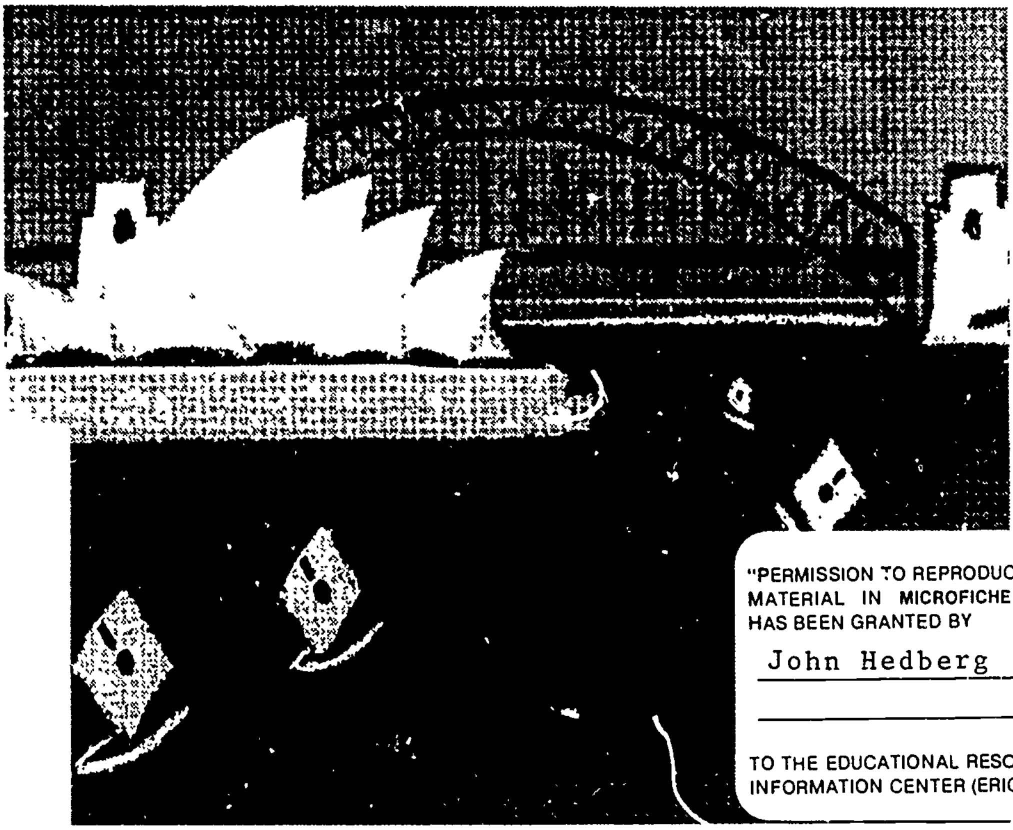
ED 305 894 IR 013 747

AUTHOR	Barrett, John, Ed.; Hedberg, John, Ed.
TITLE	Using Computers Intelligently in Tertiary Education. A Collection of Papers Presented to the Australian Society for Computers in Learning (Sydney, New South Wales, Australia, November 29-December 3, 1987).
INSTITUTION	Australian Society for Computers in Learning.
REPORT NO	ISBN-0-949088-31-5
PUB DATE	87
NOTE	477p.
PUB TYPE	Collected Works - Conference Proceedings (021) -- Viewpoints (120) -- Reports - Research/Technical (143)
EDRS PRICE	MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS	*Artificial Intelligence; *Computer Assisted Instruction; *Computer Managed Instruction; Computers; Computer Simulation; Courseware; *Distance Education; Foreign Countries; Higher Education; Instructional Systems; Interactive Video; *Media Research; Programing
IDENTIFIERS	*Australia; Intelligent CAI Systems; Intelligent Tutoring Systems

ABSTRACT

The 63 papers in this collection include two keynote addresses: "Patient Simulation Using Interactive Video: An Application" (Joseph V. Henderson), and "Intelligent Tutoring Systems: Practice Opportunities and Explanatory Models" (Alan Lesgold). The remaining papers are grouped under five topics: (1) Artificial Intelligence, including intelligent computer assisted learning, problem solving, artificial intelligence, and programming (15 papers); (2) Delivery Systems, including distance learning, communications, and hardware (9 papers); (3) Developments, including interactive video, simulation, authoring, computer managed learning, and computer based training (12 papers); (4) Research/Evaluation and Future Directions, including research, policy/planning, and philosophical aspects (21 papers); and (5) Software Tutorials, including computer assisted learning tools and commercial product applications (4 papers). The text is supplemented by various figures, and references are provided for each paper. (EW)

Using computers intelligently in Tertiary Education



Edited by John Barrett and John Hedberg

CALITE '87

Using computers intelligently in Tertiary Education

A collection of papers presented to the Australian Society for Computers in Learning, Sydney, November 29th to December 3rd, 1987

Edited by John Barrett and John Hedberg

Acknowledgements

Part of the funding for this publication has been provided by IBM Australia

Organizing Committee

Dr John Barrett, Dr Ken Wade
Mr Bill Chia, Dr John Hedberg
Ms Bronwyn Hewlett, Dr Rod Nason
Mr Rod Sims, Mr Phillip Staines
Mr Bill Stewart

Editorial Assistance
Anita de Sousa, Bill Chia, Bill Stewart

Layout and Design
John Hedberg assisted by Glen Sorensen

Conference Secretariat
Robyn Hall, Judy Layton, Mollie Lenthall

Published by the Australian Society for Computers in Learning in Tertiary Education

ISBN 0 949088 31 5

Distributed by Continuing Education, University of New South Wales, P.O. Box 1 Kensington, N.S.W. 2033.

Contents

Preface

This collection has been a learning experience for the editors, both in assembly of papers electronically and understanding the varied skills of academics in producing publishable work directly from a word processor. As we move towards a time when computers allow greater access to cheap publication, it is instructive to realize how much staff development work is still required for the writer to develop their abilities in preparing papers. Maybe one day the story of editing this collection will be told.

Under the constraints of time we have attempted the best we could, we hope the combined work of a small team of enthusiasts will have caught the errors and make interesting reading.

John Barrett and John Hedberg
Sydney
November 1987

The Marshall Harris keynote address

It takes a great deal of hard work to establish and maintain an organization such as ASCILITE. Many people have devoted a lot of energy to the success of our society. However, none of this effort or success would have been possible were it not for the initial work of Marshall Harris. Marshall was the driving force behind the first CALITE conference. It was he who took the untested concept of a tertiary oriented CAL conference and brought it to life. Many have laboured on our behalf since Marshall led the way, but they have had the advantage of the knowledge that what they strove for was viable.

In organizing that first conference Marshall had no such assurance. ASCILITE has emerged as a society from his initial input.

Marshall Harris, the members of ASCILITE thank you.

The address in 1987 was presented by Professor Alan Lesgold and a text of his presentation is included in this volume on page seven.

Patient simulation using interactive video: An application

Joseph V. Henderson, MD
Department of Preventive Medicine and Biometrics
Uniformed Services University of the Health Sciences

Training military physicians in trauma management is a dilemma in peacetime, since there are few opportunities to gain clinical experience within the military care system. We decided to attempt to use interactive video as a means of providing some experience in clinical decision making through patient simulation. Five cases, of increasing difficulty, are presented on a single videodisc side. The program is implemented on a system based on the IBM PC, and is written in the C programming language. The program emphasizes clinical realism by providing many clinical options at each decision point, and by audiovisually depicting combat clinical care in very realistic ways. The user interface is flexible and, though complex, is relatively easy to use; it is supported by a narrated, on line tutorial.

Providing surgical care to combat casualties is a problem for our armed forces. Between wars there are few military trauma patients and, consequently, few chances for military surgeons to get experience in trauma care. During wartime this shortage of experienced surgeons will be acute, and even conservative casualty projections indicate that a physician draft will not resolve the shortage. Care of combat casualties may suffer, and lives and limbs of servicemen may be jeopardized.

Part of the solution is to free experienced surgeons to do surgery, and to permit non-surgeon physicians to take care of trauma patients before and after they are in the operating room. At the early echelons of care, close to the battlefront, non specialist physicians and Hospital Corpsmen are already designated to treat patients. At the later echelons, such as combat zone field and fleet hospitals and hospital ships, our plans call for non-surgeon physicians to be assigned to resuscitate and stabilize trauma patients. Naturally, these physicians must also get training in trauma management, but clinical opportunities for this are even less likely than for surgeons' training.

As a partial solution to this problem, we are now providing Advanced Trauma Life Support (ATLS) training to active duty and reserve medical officers. ATLS was developed and is administered by the American College of Surgeons (American College of Surgeons, 1984). It's designed for surgeons and other physicians, civilian or military, who are likely to care for trauma patients. Techniques of surgery are not taught; ATLS stresses early management of trauma patients: initial resuscitation, stabilization, and disposition. The course uses a combination of lecture, small group discussion, and practical experience with animals, manikins, and volunteers simulating injury. Although physicians sometimes disagree with some of its recommendations, ATLS establishes a standard method of managing trauma patients, a standardized curriculum and content, and standards for certifying physicians in trauma management.

Simulation is particularly useful for teaching and testing the ability of students to integrate the facts and rules learned in the classroom and make patient management decisions In addition to animal and anatomic models, and live actors already used in ATLS teaching, computer-aided instruction (CAI) could also be used to simulate. CAI can instruct in an interactive way that is intrinsically motivating, and the pace and content of instruction can be tailored to the student. Using text and limited graphics, conventional CAI can present problems in a fairly realistic way. Some advantages over animal and human simulations are decreased cost and presentations of uniform content and quality.

Over the past few years, CAI developers have improved the realism of simulations by using optical videodisc technology (Leveridge, 1981; Abdulla and Henke, 1984; Woods et al, 1984). High-fidelity audiovisual sequences or still frames can be quickly accessed, depending on the instructional design and the student's choices. Computer-generated graphics can be superimposed over the video image. When used in medical education, the result can be a very realistic clinical simulation, especially useful when interaction with genuine patients is undesirable or impossible.

Interactive videodisc (IVD) is likely soon to be the preferred method for general and specialty board medical examinations. Within five years IVD will be the major testing medium for Part III of the National Board Examinations for physician certification. Recently, the American College of Cardiology has endorsed the use of computer-based IVD instruction for teaching and certification in Advanced Cardiac Life Support (ACLS) (Abdulla and Henke, 1984). ATLS has so far made no use of computer-based learning techniques, with or without videodisc.

The Naval Health Sciences Education and Training Command is now using IVD technology for medical education to train health professionals at every level. The first programs were for training Hospital Corpsmen in anatomy and physiology, and in management of emergency medical conditions. The most recent program, developed for the Navy at the Uniformed Services University of the Health Sciences, teaches ATLS principles to physicians who may be responsible for the care of combat casualties.

Hardware

The student version of the system presently uses an IBM PC XT with 640 Kbytes of random access memory, a 20 Mbyte hard disk and single 360 Kbyte floppy disk drives, a 12" Sony RGB monitor (PVM 1271Q) with light pen, a Pioneer LDV-1000 laser videodisc, and a genlock graphics card with videodisc controller (GL-512/VDC 100, Online Products Corp). The courseware authoring version of the system is as above, except that the computer is an IBM PC-AT with 20 Mbyte hard disc and 1.2 Mbyte floppy, with a monochrome monitor for programming use. Systems were purchased from Online Products Corporation, Germantown, MD.

The Sony monitor is equipped with a light pen, allowing students to enter responses without the keyboard. This can ease use of the system by less experienced students.

The laser videodisc unit is capable of randomly accessing any of 54,000 frames per videodisc side, with maximum seek time of about three seconds. Single frames such as xrays can be shown with adequate resolution. Motion sequences can run forward or backward, at normal, fast, and slow speeds. There are two audio tracks which can be included or mixed, for example, to include narrated instructions or ambient sounds to enhance realism. These features are entirely under computer control: the student's choices dictate what he sees and hears.

The GL512 'Genlock' card allows placement of computer-generated graphics over the video image. Text or pictures, with or without animation, can be used to emphasize, simulate, and produce special video effects such as windows and scrolling. The card also provides an interface giving the PC control over the videodi ~ player.

Software

Online Products Corporation provided us with IVD control routines written in assembly code. These tools drive the GL512 interfaces, and can be incorporated into author-developed computer programs ritten in the C programming language (Microsoft C, Version 4.0), to control the videodisc and computer graphics. A result is computer code that is compact and readily maintained, that executes quickly, and that is portable to other computer systems.

Unlike the PILOT authoring language, which is designed for educators who need not be computer professionals, C is a computer language designed for professional programmers: it requires detailed knowledge simply to understand a program listing. For this reason, we have developed an authoring environment written in C which allows a course author to write and display content, designate correct answers and their sequence, design screens and menus, and to control the video and audio output of the videodisc player. In this way new programs can be developed, and old ones updated, with little or no intervention by a programmer.

Instructional objectives

Our program is designed to implement ATLS goals suitable to the interactive video medium. In working through the simulated cases, the physician user will:

1. Demonstrate an understanding of the concepts and principles of primary and secondary patient assessment.
2. Show that he can establish management priorities in a trauma case.
3. Indicate an understanding of the primary and secondary management necessary within the first hour of emergency care for acute life threatening emergencies.
4. Demonstrate an understanding of proper disposition of stabilized trauma patients.

Program description

The program stresses realism: military, clinical, and interpersonal. The inherent drama of wartime medicine and casualty care is exploited to involve the learner, and draw him in a personal way into the situations presented. As in real life, the student cannot predict in advance what case he will see, even if he has used the program before. Also, doing the same thing in apparently similar situations will not always have the intended result. Finally, time is an important element, indicated by a clock displayed on the screen.

The student is presented five different cases with various combinations of airway, breathing, circulation, and neurological problems. Successive cases increase in complexity and difficulty, and they build on the experience of previous cases. The last case is a kind of final exam, having several problems.

At the usual 30 frames per second, a single videodisc side allows only 30 minutes of motion video. Disc economy requires that cases having common attributes share audiovisual sequences where possible. This means that care be taken with action, dialogue, setting, makeup, and props, to ensure continuity.

The student interacts with the program by selecting from menus either using light pen or, since all choices have unique first letters, with the keyboard. Clinical realism requires that the learner be able to select from a very wide number of choices at each decision point; it would be artificial to present a complex situation and provide only a few choices. The menu system provides 80 to 90 choices, each of which results in a more-or-less tailored feedback message indicating the correctness (within the context of ATLS) of the choice, a reason, and - if incorrect - a suggestion toward the correct choice. The menu also conveys content, since it is structured in the same way ATLS structures management of trauma; it forces the learner to think in ATLS terms in order to do the simulation. A final feature is 'windowing' and 'tiling' of menus: menus in a hierarchy are superimposed on the screen, permitting the learner to keep his orientation to that hierarchy.

Before any titles or action, there is a short tutorial to orient the student to the touch-sensitive screen and keyboard, and the menus. If the learner is working independently of a learning centre, this is always shown on the student's first use, and electively thereafter. The tutorial uses computer graphics and animation, and is supported by narration on the videodisc's second audio track.

The setting is the Casualty Receiving Area (CRA) of a 250-bed Combat Zone Fleet Hospital. The CRA is in a large tent which there are a several casualties, some injured severely, some not. There are litters and litter stands, and medical equipment and supplies needed to monitor and support the patients (IV's, AMBU bags and masks, endotracheal and chest tubes, portable Xray machines, blood pressure cuffs, and ECG monitors).

Main characters are a non-surgeon medical officer (protagonist), an experienced female nurse, a corpsman, and the triage officer, an experienced trauma surgeon. Extras are used in the background to simulate a busy CRA.

Computer-generated text and graphics overlie video footage shot on location at Ft. Meade, MD, using the Army's 10th MASH and Marine helicopter units from Quantico, VA. This establishes the combat ambience, and introduces the main characters and situation, while presenting titles. Action moves from the general to the specific, e.g, snowing groups of casualties being delivered to a field hospital by helicopter or ambulance, litter-bearers scrambling for casualties, a triage area, then into the CRA.

All the simulated cases are the same character and have the same wound, a small penetrating injury to the left anterior thorax. Possible problems associated with this wound are presented to the student, who must recognize what they are, and decide what treatment to use. Problems include upper airway obstruction, simple and tension pneumothorax, major and minor hemothorax, pericardial tamponade, abdominal hemorrhage with shock, and cervical spine fracture.

Students receive final feedback on their performance in three forms: audiovisual communication delivered by the triage officer, numeric scores, and a medical resources impact statement. The score is subjective and intended only to rank this student against others. The impact statement assesses the student's efficiency in using medical resources: if unnecessary tests or procedures were ordered, the student is told how misuse of scarce resources can result in supplies not being available for the care of other patients.

The major feedback to a care-provider is how well his patient does. Death of the patient is not presented, however. Instead, dangerous decisions by the student result in the triage officer taking over the case, which the student must then repeat. The program concludes by offering another case to the student.

We feel this program is a good example of what can be done with interactive video to simulate patient care situations. The marriage of computer and audiovisual in presenting content in realistic, clinically accurate, and dramatically absorbing ways makes for continuing medical education that can be worthwhile, rewarding, and enjoyable for the student.

Acknowledgements

Our appreciation for outstanding production support provided by Metamedia Systems, Germantown, MD, contractors for the audiovisual portion of the program, to Michael H. Spencer, Director of the audiovisual production, and to Adam R. Galper, Yale University, who designed and developed our computer programs. Our thanks to Mr. Phil Strub of the Naval Health Sciences Education and Training Command, CAMIS project manager, Mr. Hank Champagne, Production Coordinator, Naval School of Health Sciences, and to Dr. Llewellyn Legters, Chairman of the Department of Preventive Medicine and Biometrics, for his support and encouragement in the project. Thanks also to COL Brendel, Commander of the 10th MASH, Ft. Meade, MD, and MAJ Ginnie Boyle, Head Nurse, for major assistance and technical advice in the audiovisual production. Finally, we thank our board of subject matter experts, who helped to assure the clinical validity of our program: Mark Buchly, MD, FACS, MEDSTAR (Shock-Trauma Department), Washington Hospital Centre, Washington, DC; and, at the Uniformed Services University of the Health Sciences: Norman Rich, MD, FACS, Chairman, Department of Surgery, CAPT Donald Sturtz, MD, FACS, COL James Salander, MD, FACS, Department of Surgery, Eran Dolev, MD, Department of Military History, and MAJ Raymond Ten Eyck, MD, Department of Emergency Medicine.

Disclaimer

The opinions expressed by the authors do not necessarily reflect those of the U. S. Navy, the Navy Medical Department, the Naval Health Sciences Education and Training Command, or the Uniformed Services University of the Health Sciences.

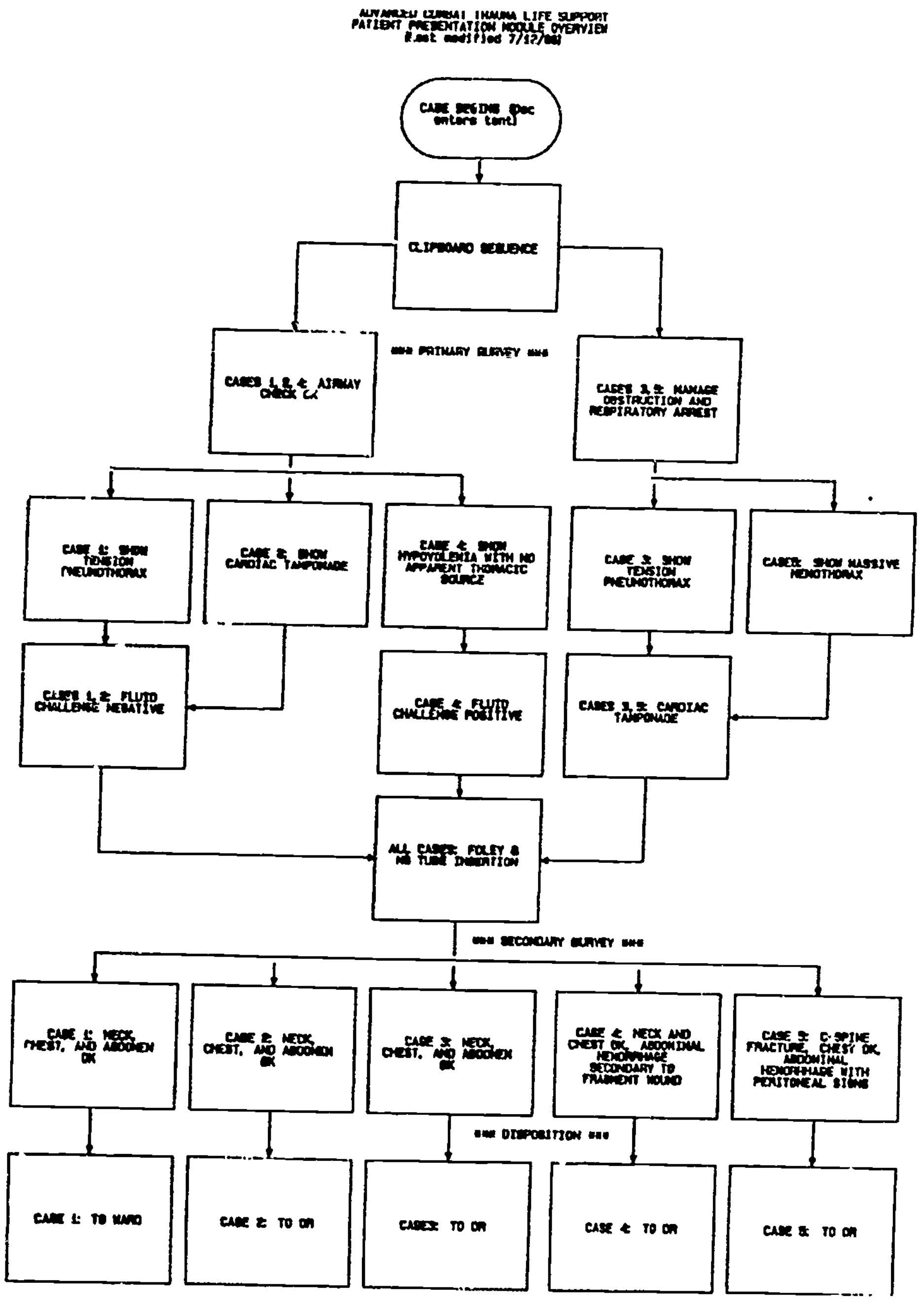
References

Abdulla, A. M and Henke, J. S. (1984). The use of natural language and a laser videodisc player in high-fidelity microcomputer-based clinical simulations. *Proceedings, Eighth Annual IEEE Symposium on Computer Applications in Medical Care*, Washington, DC, Nov, 1984, p. 936. Committee on Trauma, American College of Surgeons (1984). Advances Trauma Life Support Course Instructor Manual. American College of Surgeons.

Leveridge, L. L. (1981). Electronic processing of medical visual information. *J. Medical Systems*, 5, pp. 321-335.

Woods, J. W., Jones, R. R., Kuenz, M. A., and Moore, R. L. (1985). The optical videodisc in computer based education. *Proceedings, Eighth Annual IEEE Symposium on Computer Applications in Medical Care*, Washington, DC, Nov, 1984, pp. 937-940.

Dr. Henderson is Associate Professor at the Uniformed Services University of the Health Sciences, the U.S. federal medical school located in Bethesda, Maryland. He has joint appointments in the Department of Preventive Medicine and Biometrics and the Department of Military Medicine. He trained at the State University of New York at Buffalo and Yale University. Dr. Henderson is currently responsible for developing interactive videodisc medical training courseware for the U.S. Navy and Army. He is also conducting epidemiological research in casualty care, including development of TRAUMABASE, a multimedia data base system for trauma research and teaching which incorporates optical


ADVANCED COMBAT TRAUMA LIFE SUPPORT
PATIENT PRESENTATION MODULE OVERVIEW
(Last modified 7/12/88)
CASE BEGINS (Doc enters tent)
CLIPBOARD SEQUENCE
*** PRIMARY SURVEY ***
CASES 1, 2, 4: AIRWAY CHECK OK
CASES 3, 5: MANAGE OBSTRUCTION AND RESPIRATORY ARREST
CASE 1: SHOW TENSION PNEUMOTHORAX
CASE 2: SHOW CARDIAC TAMPONADE
CASE 4: SHOW HYPOVOLEMIA WITH NO APPARENT THORACIC SOURCE
CASE 3: SHOW TENSION PNEUMOTHORAX
CASE5: SHOW MASSIVE HEMOTHORAX
CASES 1, 2: FLUID CHALLENGE NEGATIVE
CASE 4: FLUID CHALLENGE POSITIVE
CASES 3, 5: CARDIAC TAMPONADE
ALL CASES: FOLEY & NG TUBE INSERTION
*** SECONDARY SURVEY ***
CASE 1: NECK, CHEST, AND ABDOMEN OK
CASE 2: NECK, CHEST, AND ABDOMEN OK
CASE 3: NECK, CHEST, AND ABDOMEN OK
CASE 4: NECK AND CHEST OK, ABDOMINAL HEMORRHAGE SECONDARY TO FRAGMENT WOUND
CASE 5: C-SPINE FRACTURE, CHEST OK, ABDOMINAL HEMORRHAGE WITH PERITONEAL SIGNS
*** DISPOSITION ***
CASE 1: TO WARD
CASE 2: TO OR
CASE3: TO OR
CASE 4: TO OR
CASE 5: TO OR

disc technology and the concept of hypermedia. Dr. Henderson was project director, designer, and writer for the 'Advanced Combat Trauma Life Support' program, which received the Best Overall Achievement award at the 1987 Nebraska Interactive Videodisc Competition. He lives on his sailboat, 'Green Dolphin,' with wife and two children in Annapolis, Maryland.

Intelligent tutoring systems: Practice opportunities and explanatory models

Alan Lesgold
Learning Research and Development Center
University of Pittsburgh

I present examples of a focused practice system and an exploratory environment, both implemented using artificial intelligence and graphics interface tools now becoming available. Then I describe a trouble-shooting tutor that combines these approaches. Finally, I discuss how the loose coupling of the two approaches affords opportunities for instruction in metacognitive skills.

Two of the ways in which computers can be used for education are (a) as vehicles for practice, and (b) as exploratory laboratories. Both of these modes of teaching and learning have a long history. Schools assign homework, which often consists of practice opportunities for refining knowledge that may have been presented formally in a class. Science courses, among others, often include laboratory sessions in which students learn basic principles and acquire a "feel" for a domain by exploring some of the systematic phenomena within that domain.

Both practice and exploration have posed serious problems for educators. The problem with practice is that students, left to their own devices, may end up practicing doing a procedure incorrectly — this often happens when children are given arithmetic homework, for example. The problem with exploration is that not all students are equally facile at making the principled discoveries that would justify laboratory experiences. Many of our colleagues have found it possible to overcome these problems, in whole or in part, by using the computer to present and monitor practice or exploration opportunities.

My purpose in this presentation is to show ways in which some of the methods of artificial intelligence programming and screen interface design can be applied to these two instructional approaches of practice and exploration. Then I will describe a specialized troubleshooting tutor that combines some of the properties of each approach. I will be presenting work that is, to a large extent, the work of colleagues and associates. They deserve the credit for making these systems happen, though they may not necessarily agree with the assertions I am making about their products.

Focused Practice Systems

I turn first to an example of what I will call a focused practice system. Such systems are perhaps the oldest form of computer-assisted education we have. Their purpose is to present the student with opportunities to practice material that has been introduced previously, providing feedback about the student's performance. In more recent systems, there has been an increasing effort to provide coaching or help as well as post hoc feedback. In an era of management by objectives and emphasis on the bottom line, focused practice systems seem to reflect an important component of educational philosophy. The idea is to optimize students' time by assuring that they are practicing material that they are capable of mastering but have not yet mastered.

Early focused practice systems concentrated on selecting the ri ;ht level of problems for the student and provided only binary feedback (correct or incorrect). Such systems, given sufficiently powerful prescription rules for deciding which exercises to present, have been shown to be effective for simple skill areas such as arithmetic. More recent efforts, such as the BIP system developed about a decade ago by Atkinson, have moved to more complex practice tasks, such as programming assignments, and deeper, more knowledge- driven rules for problem prescription. For the teaching of procedural skills in relatively constrained domains, focused practice environments are very powerful. Further, as you will see in the following example, they can be quite rich in the instruction they provide.

An Example: Bridge

Bridge is a practice environment for the skill of programming in Pascal (Bonar, Cunningham, Beatty, & Riggs, 1987). Rather than simply giving programming assignments, however, Bridge focuses the task of writing a program by guiding the student through three phases.

In the first phase, Bridge asks the student to state, in natural language using menus, an informal plan for the program. In the second stage, the plan is formalized by expressing it in a visual programming language. Finally, in the last phase, the student uses a structure editor to write an actual program.

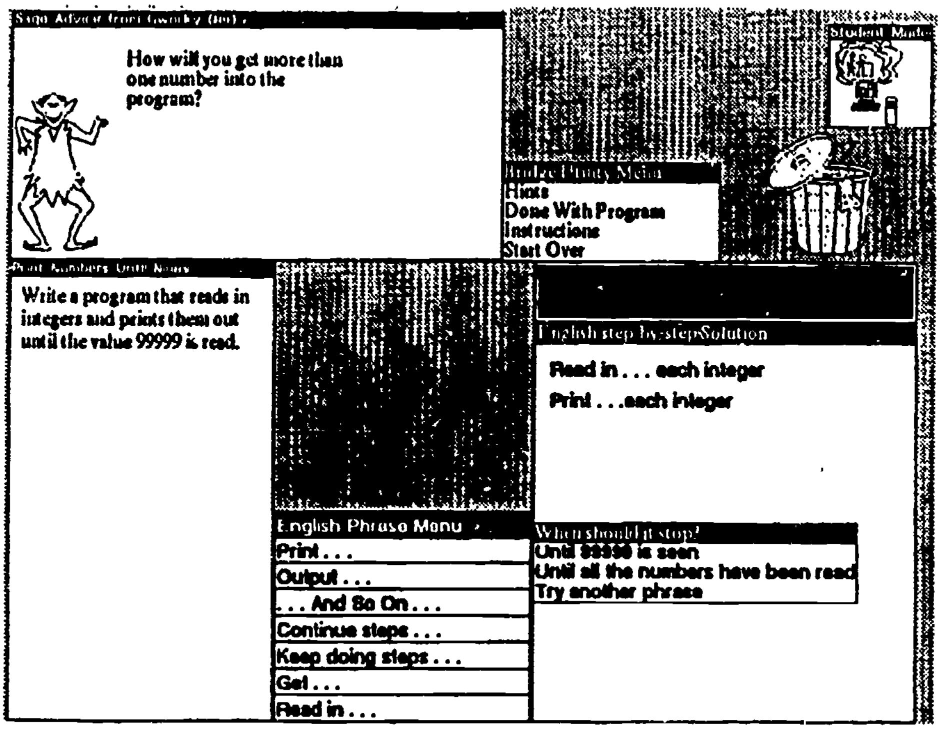
Figure 1 shows the screen display from the middle of the first phase. The problem is on the left, and the emerging student plan is on the right. Figure 2 shows the screen during the third phase. The visual plan constructed by the student is on the left, and the task is to build on the right an actual program that realizes the plan.

Let me describe Bridge's three phases briefly.

a. In the first phase, the student constructs a verbal statement describing what the program will have to do. The statement is constructed from a hierarchical menu of phrases which is based on extensive data on how students describe algorithms. The range of statements available allows for the emergence of common misconceptions about programing and the difference between a statement that might sufficiently guide a person and a description of an algorithm. For example, students can describe a loop by describing its first iteration and then saying "and so on..."

The student's product is analyzed according to the level of detail it provides and the specific components of description that are present. This two-dimensional analysis allows the tutor to say to the student that he or she has a good basic account but needs to refine it, for example, rather than simply having to list what is missing. Bridge can infer the specific naive model of the student from the description he or she constructs. Bonar's most recent work is on a formal semantics for computer program plans that will permit the procedures needed for modeling a student's knowledge with respect to a programming problem to be generated automatically (Bonar & Liffick, 1987).

b. In the second phase, the verbal account of the student must elaborate the account into a plan or model of the program's function. The components of this plan are graphic objects on the screen. Each object represents a basic building block of programs, such as a loop that terminates when a specific control value is detected for a control variable or an input routine that fetches one integer. The overall plan for the program can be thought of as a system whose components are the low-level building blocks. In order for the overall plan to be complete, the components must be connected in two ways, indicating control flow and data flow respectively. As can be seen



How will you get more than one number into the program?

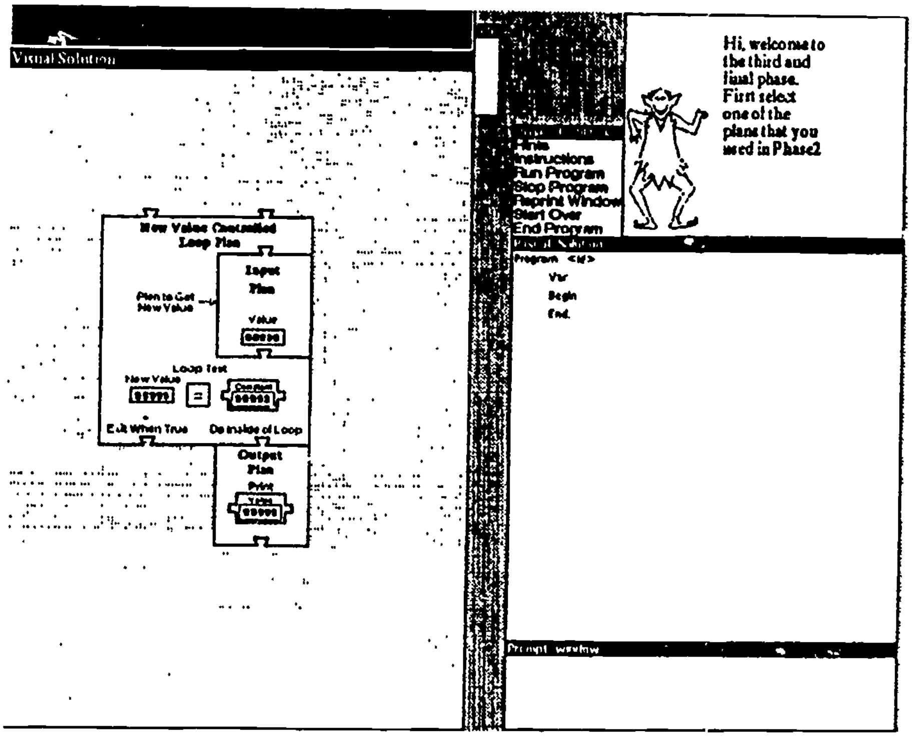
Write a program that reads in integers and prints them out until the value 99999 is read.

Read in . . . each integer

Print . . . each integer

Figure 1: Bridge Screen Display During Phase 1

Figure 2: Bridge Screen Display during Phase 2



Hi, welcome to the third and final phase. First select one of the plans that you used in Phase2

```
Program <id>
    Var
    Begin
    End.
```

in Figure 2, control flow is marked by interlocking the basic building blocks pieces, as in a jigsaw puzzle. Data flow is indicated by pointing first to a data source, such as an icon for a computed value, and then to a data port, such as the slot in an output plan that indicates what is to be output.

The second phase plan can be run as if it were a machine. When this is done, the data values actually move around, and the building blocks reverse color as they execute, so both data and control flow are illustrated. So, students finishing the second phase have a very specific mental picture, expressed overtly on the screen, of what their program will do.

c. In the third phase, the student actually writes code. Two things are important to note here. First, the student doesn't get to this point until he has a clear plan, until he and the computer agree on what the goals of the program are. Further, the student can point to various parts of his plan and thereby indicate what piece of plan any given line of code corresponds to.

Second, the usual frustrations of beginning programming, syntax mistakes, can be eliminated by providing structure editors that automatically assure the correct locations of semicolons and other Pascal exotica. In an old- fashioned classroom, syntax errors may betray either low-level mechanical failings or serious misconceptions. With the Bridge approach, such errors are inevitably mechanical. With a structure editor that allows building block shells to be selected from a menu, the syntax barrier is completely overcome.

There are important reasons for using the intermediate representations in natural language and then in the visual programming language. Consider the task faced by a programming instructor in a course. The student has a program that is incorrect, and the instructor must figure out what to say to the student. It is easy to label certain performances as wrong, but it is not so easy to communicate sufficient information to the student to assure that he will know how to do anything better next time. A good instructor will ask the student what he was trying to do and then offer advice of the form, "If you want to accomplish X then do Y." Intermediate planning languages serve the same function in Bridge. They allow the student to overtly express plans and intentions. With such languages, diagnosis in an intelligent tutor is enormously simplified.

There is a second benefit to this approach. A basic characteristic of novice problem-solving performance is that it is too much centered on the specific tactics that are available and too little concerned with understanding the problem and developing a plan for its solution. The intermediate languages of Bridge are planning languages, so the student is being channeled into a more expert approach to problem solving, in which the problem and a plan for its solution are both represented clearly before coding starts.

Overall, Bridge exemplifies three issues that are worthy of consideration in developing focused practice environments: (a) focusing on parts of the performance that are usually internal, such as an initial problem representation and a plan for solution; (b) using overt expressions by the student of plans and intentions to guide the coaching process; and (c) providing the student with tools, such as the structured editor, that maximize the proportion of time spent in significant problem solving practice.

Exploratory Systems

I turn now to an entirely different form of computer-based instruction, the laboratory environment. Laboratory environments, whether on computers or in more traditional form, are meant to deal with a specific problem. This is the large gap between the ability to verbally state a principle and any understanding of that principle. It is generally believed that if students discover a principle themselves, then they will understand it, because it is grounded in their own personal experience, whereas if they are simply told the rule, they may not clearly understand what phenomena the rule refers to. There are many experiences in education that seem to support our faith in discovery learning. For example, there are now quite a large number of demonstrations that students who are quite able to solve textbook problems in physics are not able to apply the same principles they use for those problems to everyday situations.

Discovery learning does have a problem, though. For the slower student, the wait until a discovery occurs can be very long. Regrettably, it is also the slower student who is less likely to expand beyond rote learning to actually understanding what principles of science or technology really mean. If we could somehow make discovery learning more efficient for the slower student, this would surely be a very worthwhile thing to do.

In order to tackle this goal, we need to view discovery learning as a form of problem solving. The problem to be solved is to specify the regularities present in some constrained environment, taking account of any additional constraints supplied by the teacher. Laboratory manuals generally consist of suggested activities followed by questions that are asked about those activities. Implicitly, we are hoping that students will carry out the activity and then try to understand what they observed, using the questions of the lab manual as clues. Of course, not all students do this, so labs are not always effective.

A good laboratory instructor might watch what the students are doing, examine the entries in their lab notebooks, and provide some coaching that helped students not to miss important events and to be systematic in their explorations. A computer might also be able to provide coached laboratory experiences, and there are several good reasons for using a computer for this purpose, in addition to the serious international shortage of good science teachers. Computer-simulated experiments are easily repeated and can be observed from many different viewpoints. They are safe, and they can be made relatively independent of the motor skills of the student.

How can an effective computer-based coached laboratory environment be developed? We can get some good ideas by considering an exemplary piece of work done by some other colleagues at the Learning Research and Development Center, a tutor for a simulated economics laboratory they have called Smithtown.

An Example: Smithtown

Smithtown simulates a small town's economy (Shute, Glaser & Raghavan, in press). The student can do experiments in which any of a set of factors (price, consumer preference, town population, mean income, minimum wage, weather, and number of suppliers) can be manipulated for any of a set of commodities. The effect on quantity supplied and quantity demanded is then simulated. A variety of tools are provided to the student, including a notebook/database, graph plotting tools, and a means of expressing hypotheses about the effects of various manipulations.

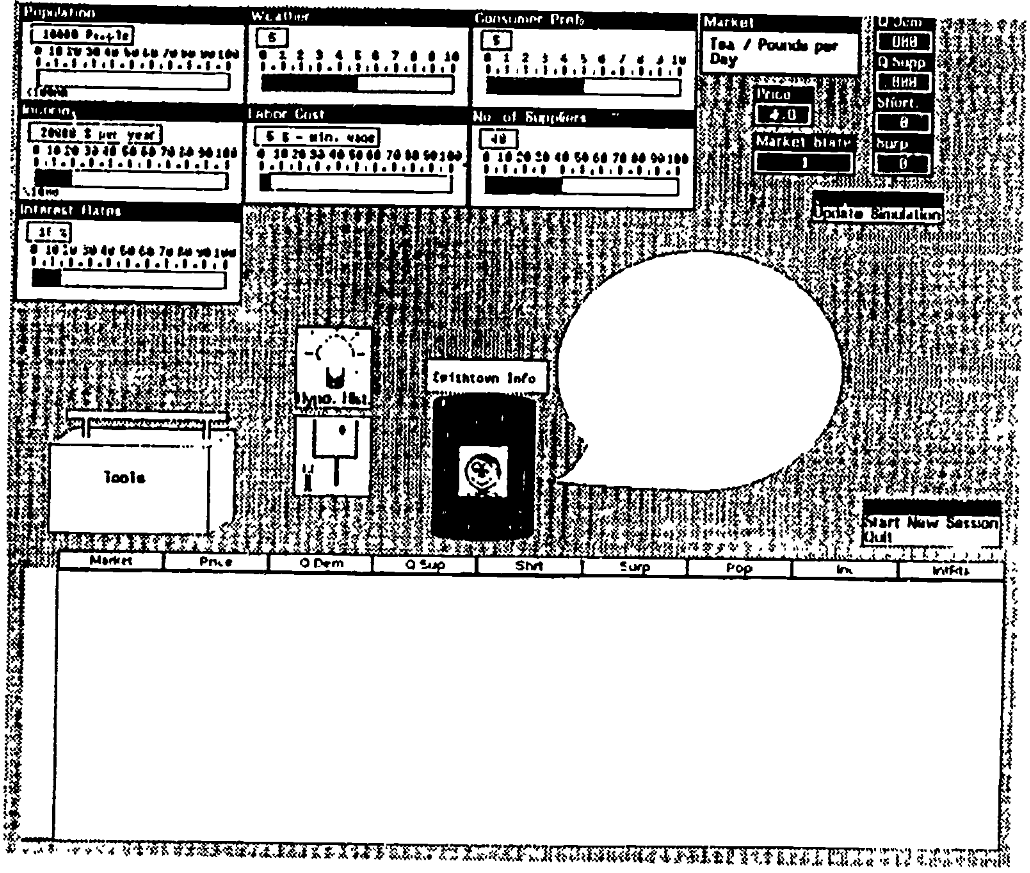
My colleagues have used Smithtown as a laboratory themselves, for studying how students use such laboratory environments. By comparing students who seem to discover principles in their experimentation in Smithtown with those who do not, and also by comparing students of differing intelligence scores, they have been able to specify a set of behaviors that the better students show when they experiment in this economics laboratory. They have then translated these findings into a set of explicit rules for coaching exploration in this laboratory setting. The current version of Smithtown, then, can watch for evidence that a student is missing some of the exploratory dispositions shown by expert learners and can then coach the student, suggesting more systematic approaches to experimentation.

I will provide three examples of the kinds of rules that Smithtown uses (Raghavan, personal communication, 1987). Each deals with a different kind of experimental skill. The first is the most straightforward. Smithtown, like many laboratory situations, has a number of variables that can be manipulated. However, it is virtually impossible to figure out what has happened if many variables are manipulated in each experiment. So, Smithtown has an internal watchdog or critic that watches to see if the student repeatedly manipulates multiple variables in a single experiment. When this happens, the following piece of advice is offered:

To really understand causal relationship (how a change affects the market), it is best to vary only one variable at a time. You have been manipulating too many variables at a time. Modify this behavior to make things easier for yourself.

A second example reflects a mixture of economic and more general strategic knowledge. If the student manipulates price of a good and then changes to a different commodity before seeing the effects of the prior price manipulation, there is a

Figure 3: A Sample Screen from Smithtown



Population: 10000 People
Weather: 5
Consumer Prefs: 5
Income: 20000 $ per year
Labor Cost: 5 $ - min. wage
No. of Suppliers: 40
Interest Rates: 10 %
Market: Tea / Pounds per Day
Price: 4.0
Market State: 1
Q Dem: 088
Q Supp: 088
Short.: 0
Surp: 0

Market	Price	Q Dem	Q Sup	Shrt	Surp	Pop	In	IntRts

possibility that the price change in the first good might influence the price of the second good without the student even realizing this. So, the student doing something like this is given an appropriate warning.

The third kind of criticism is provided when the student's verbal hypotheses don't match well with his experimental behavior. For example, a good hypothesis should state an effect on a dependent variable that is produced by some manipulation of one or more independent variables. If a student claims that changes in a dependent variable produce some effect, this is worthy of coaching comment, and Smithtown provides this coaching. Overall, the coaching aims at differences of the following kinds, that were found between faster and slower learners who used Smithtown:

1. Generating and testing hypotheses.
2. Engaging in richer experimentation within a single market setting rather than randomly changing markets.
3. Replicating experiments to be sure that they knew what produced the effects.
4. Changing one variable at a time, holding everything else constant.
5. Making big enough changes so that effects could easily be noticed.
6. Generalizing hypothesized effects across markets.
7. Having systematic notebook skills.

Overall, then, Smithtown has provided evidence that skills associated with lesser learning ability can be tutored by a computer-based laboratory. An important question is whether the effects are local or general. That is, does the coaching merely improve the learning of economics from the economics tutor, or do the coached exploratory skills generalize to other laboratory situations. To test for this, my colleagues have developed two other exploratory environments, for simple electrical principles and for geometric optics, so they can begin to examine transfer between domains.

Getting the Best of Both Worlds

We've now considered two very different computer-based instruction paradigms, focused problem solving practice and laboratory exploration. Each contained a little bit of the other. For example, the second phase of Bridge, the visual programming phase, is partly a laboratory. One builds and runs program plans. Similarly, one can imagine making the exploratory economics environment quite directive, with problems like "What happens to quantity supplied as price rises?" However, we have not yet seen a systematic effort to combine exploratory environments with focused problem solving, and the design issues raised by such a combination have not been fully explored.

An instructional system my own laboratory at LRDC has been working on provides some opportunities for reflection on how the two approaches might be combined and on what concerns arise in such hybrid designs. We have been developing a tutor for a specialized electronics troubleshooting job in the U. S. Air Force. The people our tutor will train are responsible for repairing navigation equipment for a fighter plane. They have lots of troubleshooting guides and test equipment for doing this, and they do it well.

However, the test station, forty cubic feet of electronics, that they use in their work also breaks from time to time, and fixing it is a lot harder than using it to fix parts from planes. Our tutor provides a partial simulation of this test station and also poses specific diagnosis problems. It provides advice and focuses problem solving activity considerably, but it is also a reasonably rich exploratory environment, and we could make it richer. For example, within limits, we could probably produce an instructional system that would allow a student to ask for a simulation of the effects of any of a large set of component failures on overall test station performance.

This project started when we were asked to take on a primary role in developing and testing methodology for a large Air Force effort to improve the quality of training and performance measurement for a number of their more technical occupational specialties. Specifically, we were asked to explore the feasibility of using cognitive psychological approaches to task analysis (Lesgold, Lajoie, et al., 1986). As that work proceeded and cognitive task analysis approaches were developed, we were also asked to demonstrate the efficacy of our analytic approaches by showing that skills we identified as critical and not uniformly acquired would, if taught, make a difference in airmen's job performance. This latter effort is now underway and involves development and field testing of an intelligent tutoring system.

To carry out this project, we have had to develop cognitive task analysis techniques that we believe are of broad general interest. In particular, we have developed a combination of structured interview and think-aloud protocol that seems quite generally useful. Several other approaches we have developed (Cf. Lesgold, Lajoie et al., 1986) also seem likely to be useful. Further, we have found it beneficial to compare subjects at multiple levels of competence, so that we can describe competence as the presence or absence of certain key features during realistic (job-like) problem solving performances. The features, in turn, signal specific procedural knowledge that seems to be differentially acquired by those who learn well on the job. Our current task is to try, via intelligent coached practice in a simulated job environment, to provide this procedural knowledge and related conceptual (declarative) knowledge to those airmen who do not already have it. Let me take just a minute or two to describe the structured interview technique, since I think you will find it useful.

Structured Interviews during Problem Solving: The Concept of Effective Problem Space

In our first effort to develop a task analysis scheme that would be sufficient to inform development of an intelligent instructional system, Drew Gitomer developed a troubleshooting task that involved detection of complex faults in the test station used by our subjects — prior Air Force data suggested that this was their hardest job. As a first formative approach, he simply videotaped subjects attempting to solve such fault detection problems. He then examined the protocols (transcriptions of the tapes) and attempted to count a variety of activities that seemed relevant to metacognitive as well as more tactical aspects of problem solving in this domain. While the results, published in his thesis (Gitomer, 1984), were of great interest, we wanted to move toward a testing approach that was less dependent upon skilled cognitive psychological training. That, after all, is one aspect of what test development is largely about — rendering explicit the procedures that insightful researchers first apply in their laboratories to study learning and thinking. Also, we felt that in a technical skill, much of a person's capability is very much grounded in the specifics of the task (cf. Scribner, 1984). Thus, we expected a method more heavily driven by domain expertise to be more productive than a "weak method."

In fact, our breakthrough came not so much from our psychological expertise but rather from interactions of the three of us, who had substantial cognitive psychological training, with an electronics expert, who had extensive experience watching novice troubleshooting performances. He pointed out that it was quite possible to specify the entire effective problem space, even for very complex troubleshooting problems. That is, he could pretty completely tell the rest of us all of the steps that an expert would take as well as all of the steps that any novice was at all likely to take in solving even very complex troubleshooting problems. In this case, the task was to find the source of a failure in a test station that contained perhaps 40 cubic feet of printed circuit boards, cables, and connectors, but various specific aspects of the job situation constrained the task sufficiently so that the effective problem space could be mapped out.

This then created the possibility that we could specify in advance a set of probe questions that would get us the information we wanted about subjects' planning and other metacognitive activity in the troubleshooting task. For what is probably the most complex troubleshooting task we have ever seen, there are perhaps 55 to 60 different nodes in the problem space, and we have specific metacognitive probe questions for perhaps 45.

Figure 4 provides an example of a small piece of the problem space and the questions we have developed for it.

An examination of the questions in the Figure reveals that some are aimed at very specific knowledge (e.g., How would you do this?), while others help elaborate the subject's plan for troubleshooting (consider Why would you do this? or What do you plan to do next?). Combined with information about the order in which the subject worked in different parts of the problem space, this probe information permits reconstruction of the subject's plan for finding the fault in the circuit and even provides some information about the points along the way at which different aspects of the planning occurred. In fact, we went a step further and also asked a number of specific questions about how critical components work and what their purpose is. Finally, when a subject was headed well away from a reasonable solution path, we would, at preplanned points, redirect his efforts back to more fertile ground.

In more recent work, we have standardized a set of questions that are asked at each node of the problem space. These questions address the precursor assumptions implicit in reaching a given point of the problem space, the action to be taken at that point, the expected results of that action, and the interpretation of the results both in terms of a model of the device and in terms of appropriate next steps. We call this the PARI technology, because it focuses on the Precursor, Action, Result, and Interpretation at each node of the problem space.

Christopher Roth and Gary Eggan have taken our methods and applied them to a slight variant of the domain we studied. However, they spent more of the total effort comparing the specific problem-space paths of experts and novices. The next few figures illustrate the yield of this work. Roth worked with Eggan to develop a collection of diagnosis problems that were characteristic of real problems faced in the work environment and that were likely to be solved correctly by the best technicians and not by many of the less-skilled, first-term airmen. People with experience in handling the tasks on which many airmen have difficulty tend to have a good sense of the kinds of problems that are in this class (representative of the domain, solvable by the better technicians, and not solvable by a substantial number of workers), and Eggan's advice has proven itself empirically.

Figure 5 illustrates the effective problem space that Eggan helped Roth construct for the problem. It was felt that the bulk of problem solving activity would fall within this space, for both experts and less-skilled workers.

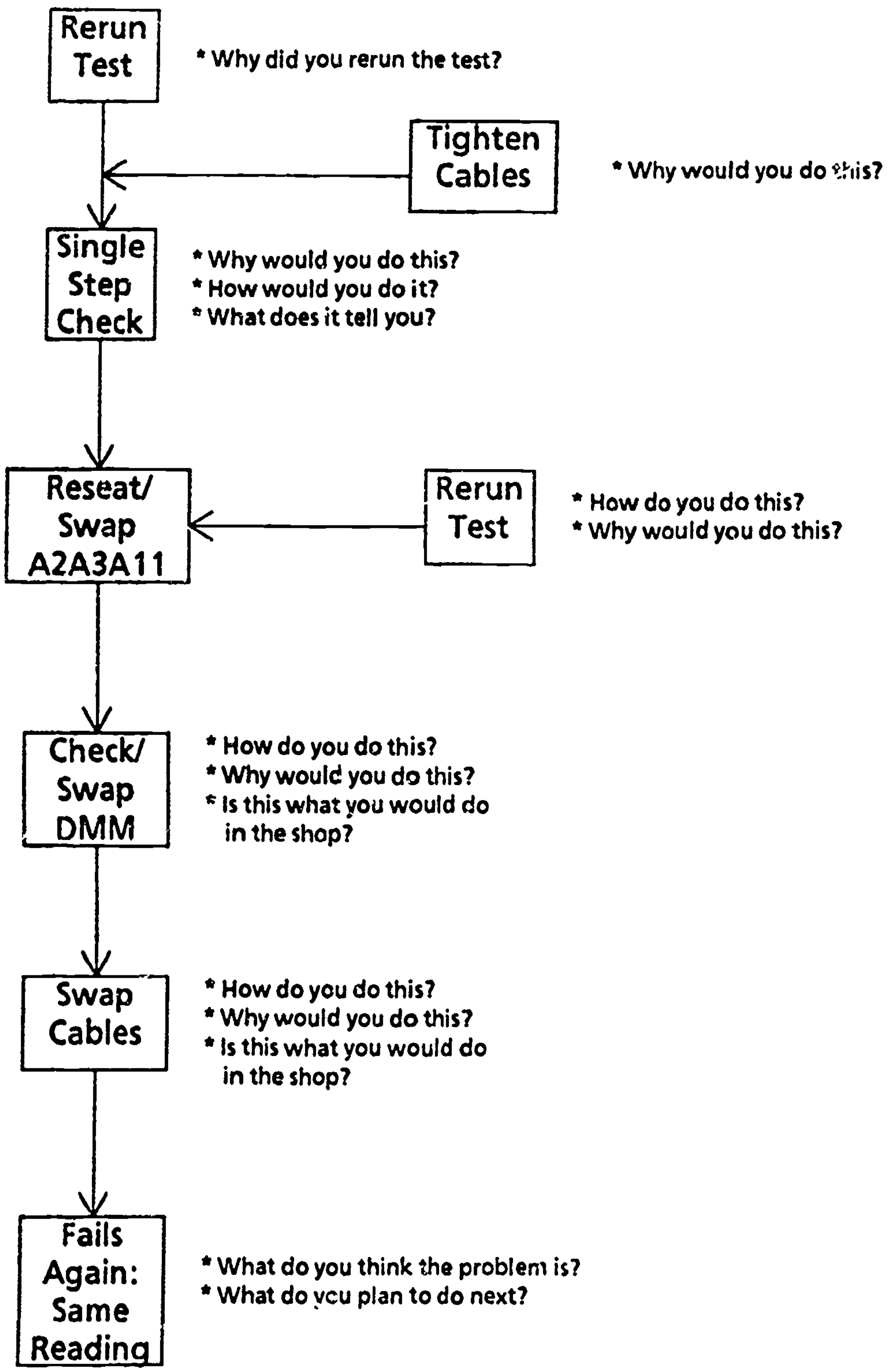


Figure 4: Effective Problem Space Methodology — First Version

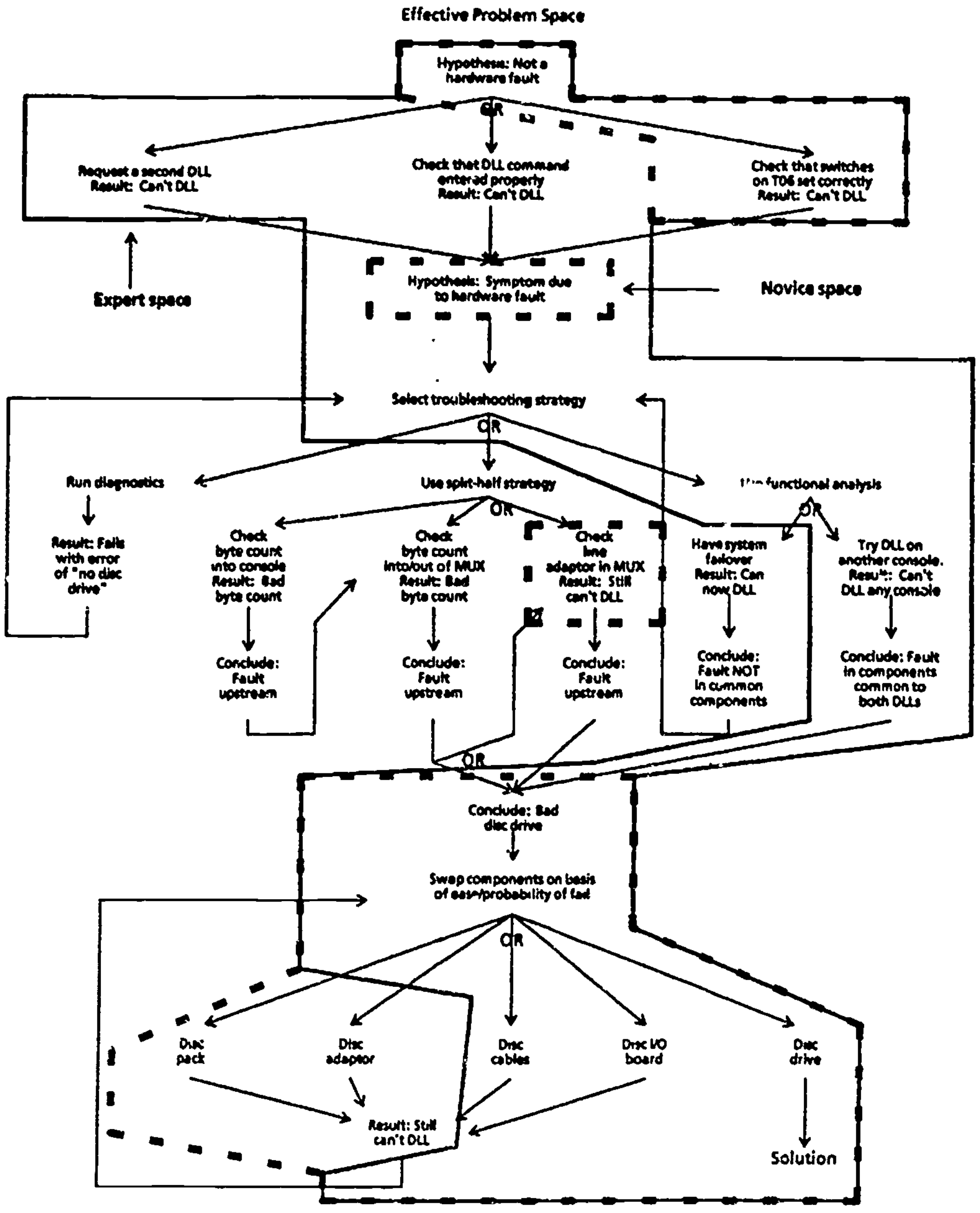

Effective Problem Space
Hypothesis: Not a hardware fault
OR
Request a second DLL Result: Can't DLL
Check that DLL command entered properly Result: Can't DLL
Check that switches on T06 set correctly Result: Can't DLL
Expert space
Hypothesis: Symptom due to hardware fault
Novice space
Select troubleshooting strategy
OR
Run diagnostics
Result: Fails with error of "no disc drive"
Use split-half strategy
OR
Check byte count into console Result: Bad byte count
Check byte count into/out of MUX Result: Bad byte count
Check line adaptor in MUX Result: Still can't DLL
Conclude: Fault upstream
Conclude: Fault upstream
Conclude: Fault upstream
functional analysis
OR
Have system failover Result: Can now DLL
Try DLL on another console. Result: Can't DLL any console
Conclude: Fault NOT in common components
Conclude: Fault in components common to both DLLs
OR
Conclude: Bad disc drive
Swap components on basis of ease/probability of fail
OR
Disc pack
Disc adaptor
Disc cables
Disc I/O board
Disc drive
Result: Still can't DLL
Solution

Figure 5: Effective Problem space Segment, Showing Expert-Novice Differences

Roth and Eggan then proceeded to run subjects on this task. The problem was posed verbally to airmen who had access to the full set of Technical Orders (documentation for the devices involved). Technicians were encouraged to offer their hypotheses, state their plans, and announce specific steps they would take in attacking the problem. Whenever an overt step was stated by a subject, he was immediately informed of the outcome of that step.

Figure 5 also shows the basic results Roth and Eggan obtained. The experts, whose performance is outlined with a solid line, used a subset of the effective problem space, avoiding some of the steps that were less likely to be productive. The novices, whose empirical problem space is outlined in heavy dashes, failed to make some of the expert moves and made some moves that experts would not make, such as swapping a disc pack (see bottom left of Figure 5). Further, their empirical problem space was discontinuous, a set of islands. To better understand what knowledge separated the experts from the novices, Roth and Eggan then tried to determine the specific knowledge that experts used at critical points in the problem space. Knowledge involved in making a move that experts make and novices do not is a clear candidate for additional training.

A final issue that Roth and Eggan addressed was the matter of the discontinuities in novice problem spaces. In essence, the gaps represent impasse points where the novice does not know what to do. At those gaps, we see relatively random behavior, driven by the snippets of knowledge the novice has. For example, in the case shown in Figure 5, there is initially a period of board swapping that is not directed specifically at the problem. Then, there is an effort to think about the path involved in a download. The multiplexor (MUX) somehow becomes the focus of attention, and thought about the MUX continues until the novice is redirected toward the disc drive, whereupon the right possibilities are addressed, but without concern about the relative costs and benefits of the five alternatives.

From Testing to Tutor Development

The Effective Problem Space PARI technology gives us two things we need in order to build an intelligent tutor: a technology for deciding what to teach; and a technology for constructing individual problems for the tutor to present. The decisions on what to teach come from comparisons between effective and less effective workers in the job environment. In the case of our tutor, such comparisons showed us that we needed to focus on tracing of signals through schematic diagrams, the more sophisticated uses of test equipment, mental models of the process, within the test station, of carrying out a test of an aircraft component, and domain-specific customizations of some general troubleshooting strategies. We then put our subject matter experts to work generating problems that treated the issues we had identified and that involved various parts of the test station. For each problem, we have an effective problem space data structure that contains all the information needed to interact with the trainees as they attempt to solve the problem.

Each node of the effective problem space for a problem is represented as an instance of a computational object. When the trainee chooses to move to a node of the problem space, its corresponding object receives a message telling it to take charge of interactions with the trainee. The object has the ability to generate certain kinds of hints and contains specific text for other hints that might be needed at that point. It knows what actions can be performed at this point, what prerequisites those actions might require (e.g., shutting off power before removing a card from the station, getting the right schematic for circuit tracing, etc.), what the results of those actions are and what they mean, and what options to present to the trainee as the next step. Each node also contains a list of the curriculum goals relevant to it.

This curriculum goal information lays the basis for the student modeling activity that is needed to assign problems to the trainee and to decide how supportive to be in offering help at various points in the trainee's efforts to solve a problem. The student modeling process we use is still very primitive. When a student starts a problem, an estimate is made of his likelihood of successfully negotiating each node of the problem space. This estimate is made on the basis of a long-term student model that shows the learning state of each curriculum issue If the student has mastered the issues relevant to a node, then we can expect him to do well at that node and can therefore give only subtle hints. On the other hand, if we know a node requires knowledge the student has not exhibited, we should provide more explicit help. Our categories are currently simple: Good, OK, and Bad are the possible levels of expectation for any given node. We call the expectation information that is generated for a specific problem when a trainee attempts it the performance model of the student for that problem. After the problem is completed, deviations of the performance model from actual student performance (e.g., did he breeze through a spot where he was expected to have trouble, or did he ask for lots of hints at a spot he should have been able to handle) are used to update the ratings assigned to each curriculum issue for that student. We call these ratings the competence model of the student.

So, we assign a problem by matching problem characteristics with our current assessment of what the student knows how to do, we generate an expectation of his performance at each point in the problem space by noting what curricular issues are relevant at that point and what we know about the trainee's competence on those issues, and then, after the problem is completed, we update the competence model to reflect unexpected aspects of trainee performance as he went through the problem.

Adding an Exploratory Component

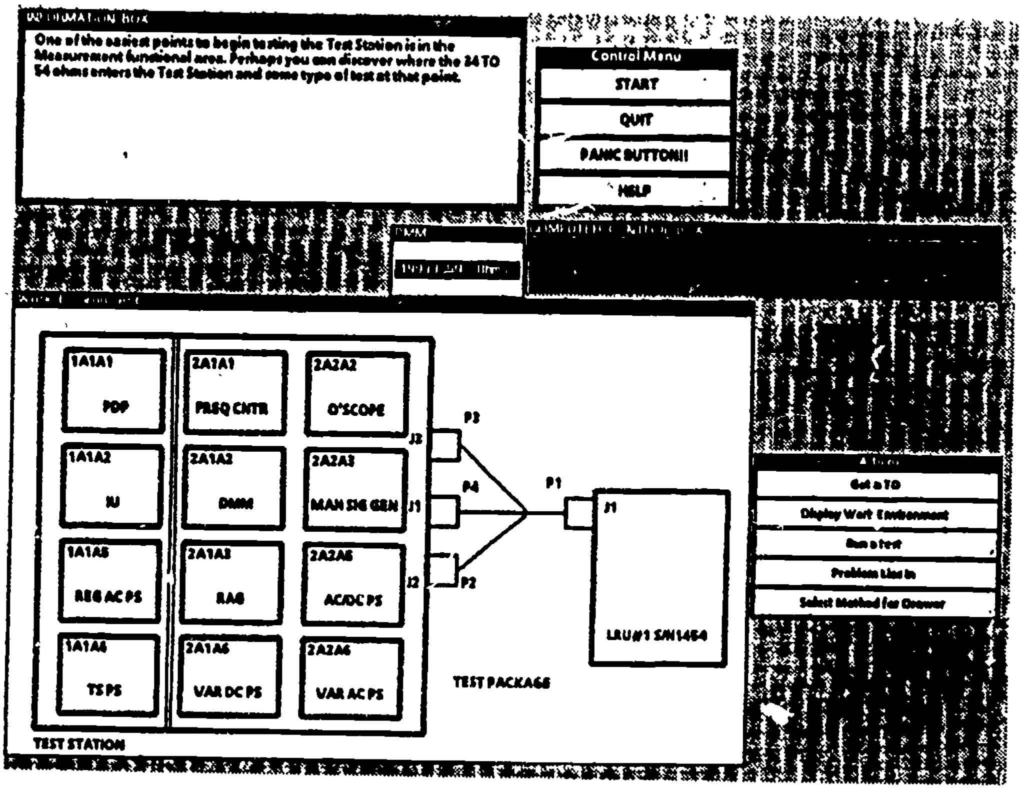
So far, what I have described is a focused problem solving tutor. We give the student a problem and, as necessary, provide hints that keep him focused on its solution. The next step was to add an exploratory component that permitted the student to try out some of his ideas. We did this by adding a simulation of the front panels of the test station and an ability to conduct simulated tests of the test station circuitry by placing meter probe icons at various points in schematic diagrams of the innards of the test station.

A few pictures are worth many words here. Figure 6 shows the screen in the midst of a problem. A representation of the front of the test station is shown. Pressing the mouse button in any of the 12 boxes on the left would cause the front panel corresponding to that part of the test station to appear. The choice of how to proceed from the current problem space node is made by selecting from the menu on the right. The picture was taken right after the student asked for a hint, which can be seen in the information box in the upper left.

Figure 7 shows the screen after one of the front panels has been requested. The student can move knobs by mousing the setting he wants. When knobs are moved, meters automatically update. The student can also request documentation on the screen.

Figure 8 shows a page of the Technical Orders, which are the documentation for the test station. They are accessed via an index on which the student can mouse-button for the entry he desires.

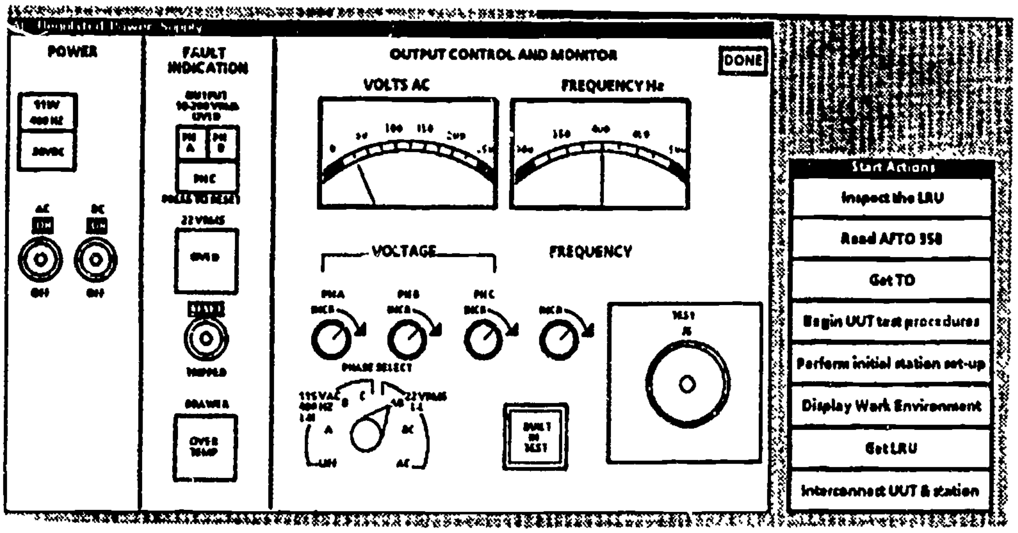
Finally, Figure 9 shows a schematic diagram. The student can ask for measurements of voltage, current, or resistance values by calling up an icon of a meter, setting its front panel, and then indicating where its probes should be placed in the schematic diagram. The upper left of the schematic has above it the icon for a Simpson multimeter, and its red and black probes have been placed near the middle of the schematic on the left (the red probe

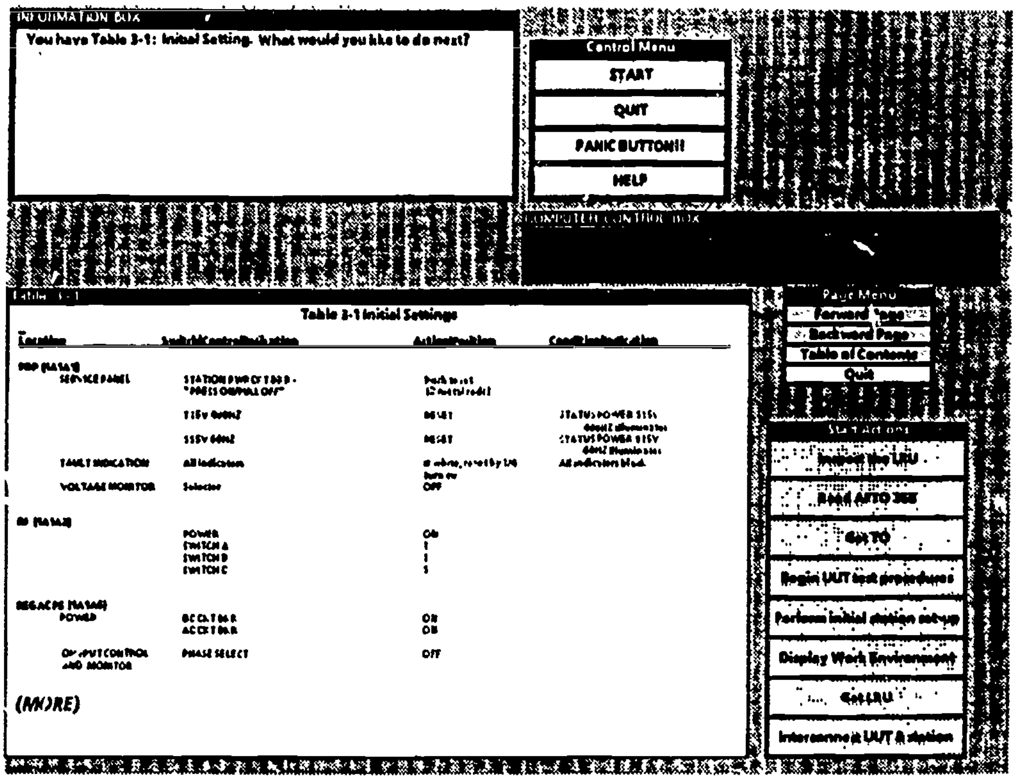

INFORMATION BOX

One of the easiest points to begin testing the Test Station is in the Measurement functional area. Perhaps you can discover where the 34 TO 54 ohms enters the Test Station and some type of test at that point.

Figure 6: Sherlock Screen Showing a Hint Being Given

Figure 7: Sherlock Screen Showing a Manipulable Panel





You have Table 3-1: Initial Setting. What would you like to do next?

Table 3-1 Initial Settings

Location	Switch/Control/Indication	Action/Position	Condition/Indication
SDP (1A1A1) SERVICE PANEL	STATION POWER TEST - "PRESS ON/PULL OFF"	Push to set [illegible]	
	115V 400HZ	RESET	STATUS POWER 115V 400HZ illuminates
	115V 60HZ	RESET	STATUS POWER 115V 60HZ illuminates
FAULT INDICATION	All indicators	[illegible]	All indicators blank
VOLTAGE MONITOR	Selector	OFF	
[illegible] (1A1A2)	POWER	ON	
	SWITCH A	1	
	SWITCH B	1	
	SWITCH C	1	
REGAC PS (1A1A4) POWER	DC CKT BKR	ON	
	AC CKT BKR	ON	
OUTPUT CONTROL AND MONITOR	PHASE SELECT	OFF	

(MORE)

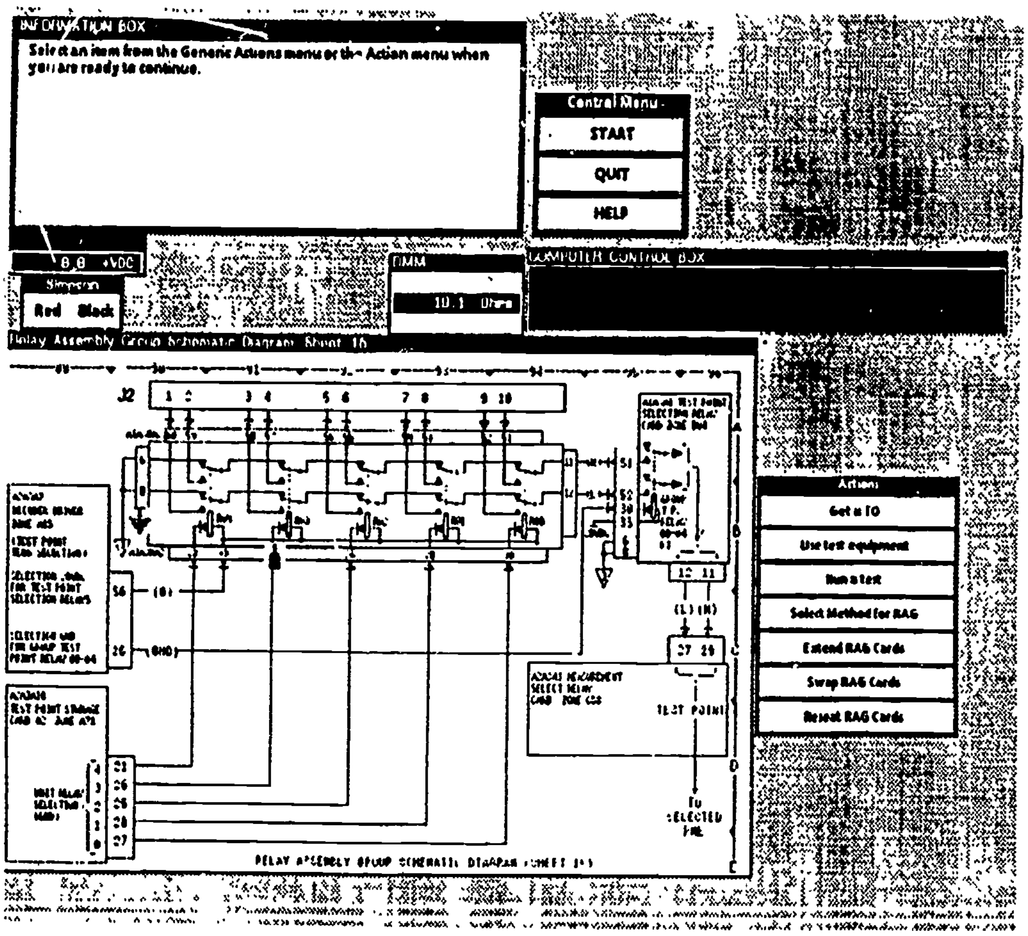
Figure 8: Sherlock Screen Showing a Page of Documentation

looks like a meter probe, and the black probe looks like the symbol for ground). Obviously, there are test point combinations for which we will not be able to provide result values, but the number of test values seems to be adequate so far.

Coupling the Exploratory Environment and the Focused Problem Solving Tutor

The remaining task is to couple the exploratory environment, the simulation of the test station, with the menu-driven focused problem solving tutor. This is done in a very simple way. At some, but not all, points in the course of problem solution, the menu includes the option of doing a test. When the trainee takes that option, he gains access to the exploratory environment. The tutor can control what explorations he is permitted, allowing either general exploration or restricting exploration to a specific circuit component. When the trainee tries to perform a specific test, he must, of necessity, have in front of him the schematic that shows the points at which he wants to make a measurement. When he tries to make a measurement, the object in charge of that schematic can impose special conditions, such as turning off the power, before the test is permitted. Similarly, the object representing the test instrument may impose conditions on how it can be set. When all conditions are satisfied and the probes are placed, the meter icon is automatically updated to show the test value.

The system keeps a record of all tests it performs. When the student indicates that he is done performing tests, a miniproduction system attached to the schematics on which tests have been performed is al-


INFORMATION BOX

Select an item from the Generic Actions menu or the Action menu when you are ready to continue.

Figure 9: Sherlock Screen Showing a Testable Schematic

lowed to execute. It contains rules that show the relationship between various problem-space activities and various tests. A rule might say that if certain parts of the problem space have already been negotiated and certain tests are performed, then a large portion of the test station is ruled out as the cause of the fault being diagnosed. The problem space nodes relevant to work in the ruled-out portion are notified that they are no longer necessary steps. Later, if the student enters ruled-out portions of the problem space, he may be allowed to proceed but will eventually be told that he really had ruled out those possibilities when he was performed a test earlier (the test can be described, of course).

So, with a relatively simple convention, a production system that inter-relates exploratory actions with the representation of the problem space, we can couple the two approaches. We currently write productions by hand to do this, but it is probably possible to build intelligent components to do this automatically, assuming that the semantics of each problem space object are part of its knowledge. One would simply ask the objects relevant to the part of the test station that had been explored to decide whether the tests that were done were relevant to themselves. When an object decided that a test was relevant to itself, it would, given an appropriate method, then decide what the effect of conducting the test should be on the relevance of that point of the problem space for further search.

On the Value of Not Being Perfect

I conclude by noting that the techniques we have used are provably incomplete. Theoretically, there are actions that a trainee could want to take that are not represented in our effective problem spaces, and there are sensible tests one could perform that our experts might not anticipate. The hypothesis we are now testing by taking our tutor to the field is that such imperfect approaches can still provide lots of useful instruction. It is not clear that a complete simulation of a device as complex as the test station we are working with can be developed at an affordable price. Further, perfection, while it may be comforting to the designer, may not be necessary to effective instruction. We see ourselves as involved in an activity similar to that of many industrial designers, designing a system which is not perfect but which can accommodate, through a variety of mid-course corrections, to any problems for which it was not fully prepared.

One of our accommodation techniques, the panic button, has actually turned out to be quite interesting as a new source of instruction. When the tutor completely loses track of where the trainee is in the problem space, perhaps because the set of tests the trainee performed in the exploratory environment was pretty random, it tells the student this. Also, when the student is stumped and the hints he is getting are not helpful, he can hit the panic button. In panic mode, the tutor works with the student in a top-down manner, discussing what the student has done so far and what parts of the problem space remain unsearched. It then asks the student to proceed from the top, specifying which subspace and which subsubspace, etc., he wishes to enter. Put another way, when the tutor doesn't know what to do, it carries out, and thereby demonstrates, a systematic set of metacognitive strategies that the trainee would do well to model.

Heidegger observed that for the most part we are controlled by our recurrent, everyday, expected environment. Only when breakdowns occur in the expected situation do we have to think about what to do. Only then can we learn. An interesting accident of the approach we have taken in designing our tutor is that we cannot avoid some breakdowns of the tutoring process. Upon reflection, these breakdowns may be the best opportunities for teaching that our tutor has, and the default techniques it can apply may be useful techniques for the trainee to encounter and to see demonstrated.

In the complex interdisciplinary world of computer technologies for education, I expect to make many mistakes, to seldom produce perfect instruction. Consequently, I look for ways to build on mistakes and insufficiencies because I can count on having them. At the moment, it looks as if this is a fruitful way to proceed.

References

Bonar, J., Cunningham, R., Beatty, P., & Riggs, P. (1987). Bridge: *Intelligent Tutoring with intermediate representations*. Technical Report. Pittsburgh, PA: University of Pittsburgh, Learning Research and Development Center.

Bonar, J. G., & Liffick, B. W. (1987). *A visual programming language for novices*. Technical Report. Pittsburgh, PA: University of Pittsburgh, Learning Research and Development Center.

Gitomer, D. H., (1984). *A cognitive analysis of a complex troubleshooting task*. Ph.D. Dissertation. Pittsburgh, PA: University of Pittsburgh.

Lesgold, A. M., Lajoie, S., Eastman, R., Eggan, G., Gitomer, D., Glaser, R., Greenberg, L., Logan, D., Magone, M., Weiner, A., Wolf, R., & Yengo, L. (1986, April). *Cognitive task analysis to enhance technical skills training and assessment.* Technical Report. Pittsburgh, PA: University of Pittsburgh, Learning Research and Development Center.

Scribner, S. (1984). Studying working intelligence. In B. Rogoff & J. Lave (Eds.), *Everyday Cognition: Its Development in Social Context.* Cambridge: Harvard University Press.

Shute, V., Glaser, R., & Raghavan, K. (in press). Discovery and inference in an exploratory laboratory. In P. L. Ackerman, R. J. Sternberg, & R. Glaser (Eds.), *Learning and Individual Differences.* San Francisco: Freeman.

Alan Lesgold graduated from Stanford University in 1971 with a PhD degree in psychology. Since then, he has been at the University of Pittsburgh's Learning Research and Development Center, where he is currently Professor of Psychology and Associate Center Director. With Richmond Thomason, he directs the University's doctoral program in intelligent systems studies, which focuses on formal characteristics of intelligent organic and artificial systems. His laboratory has been developing prototype computer-based training systems that are based upon cognitive psychological principles and artificial intelligence techniques. These intelligent training systems also function as tools for instructional research. Domains for which systems are being developed include electronics troubleshooting, programming, flight engineer duties, electrical and logic principles, economics, and composition (writing). Lesgold has been involved in a number of efforts to plan research strategies on applications of computers to education, both in the United States and elsewhere.

The Physics literacy project — 1987

D.E. Bailey*, P.F. Logan**, P.J. Walker**, I. Walsh
* School of Science and Technology, Nepean C.A.E.
** Physics Department, University of Technology

In 1986 the Commonwealth Tertiary Education Commission, through the Participation in Equity Program, supported the Physics Literacy Project. This was a proposal to devise a number of computer assisted learning packages to assist students with little or no background in Physics, to overcome the initial hurdles to a Tertiary Physics course. As the program matured this had developed into seven CAL programs using the Macintosh computer and the programming language Forth. The project has a strong emphasis on experiential learning of basic concepts using simulations. This paper describes the progress of the project and will present a number of issues relevant to the production of CAL material and particularly CAL material using simulations.

The Physics Literacy Project was reported on at CALITE - 85 (Bailey 1985) indicating the origins of the project from a concern over students who were attempting tertiary Physics programs with a lack of secondary background in Physics. This concern is supported year after year at many institutions by a continued attrition or high failure rate amongst these students (80 - 95 %)

One approach towards assisting these students is to produce Computer Assisted Learning (CAL) material. This project selected 7 topic areas and work has proceeded towards producing teaching material for the Macintosh computer using Forth as a programming language. The justification for the computer choice was based on the interaction possible via mouse between operator and machine which (a) reduces the need for keyboard skills by the student and (b) gives the student a "hands-on" physical experience of controlling directly the on-screen events. Despite difficulties experienced with the language and its implementation on our basic 512 K Macs and the lack of acceptable large screen projection facilities we are still delighted with our choice of machine and method of operation.

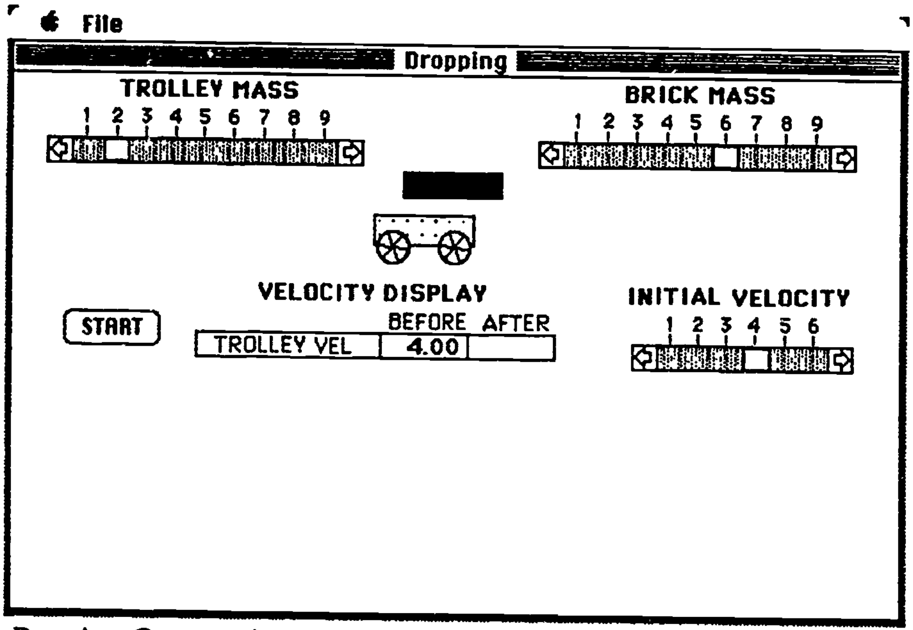
The Project

The initial proposals emphasised the team design of software modeled on Bork (1984). This has had mixed success. For the team members the process has been very rewarding as it has forced them to go back to basics and to examine seriously their teaching approaches. However some designers preferred to work as individuals and some teams lost members due to commitments outside the project.

It was also found that the revision and design process needed feedback from using the machine and this was not always available. Due to the quaint way funding works, the project has had two programmers, both on one year contracts. Part of their time has been spent learning the machine and language before producing product. Early work has concentrated on the production of graphic sequences towards later incorporation into teaching sequences. This has resulted in a number of exciting mini programs of great value towards supporting a teaching program but unfortunately not specifically appropriate to the immediate goals of the project.

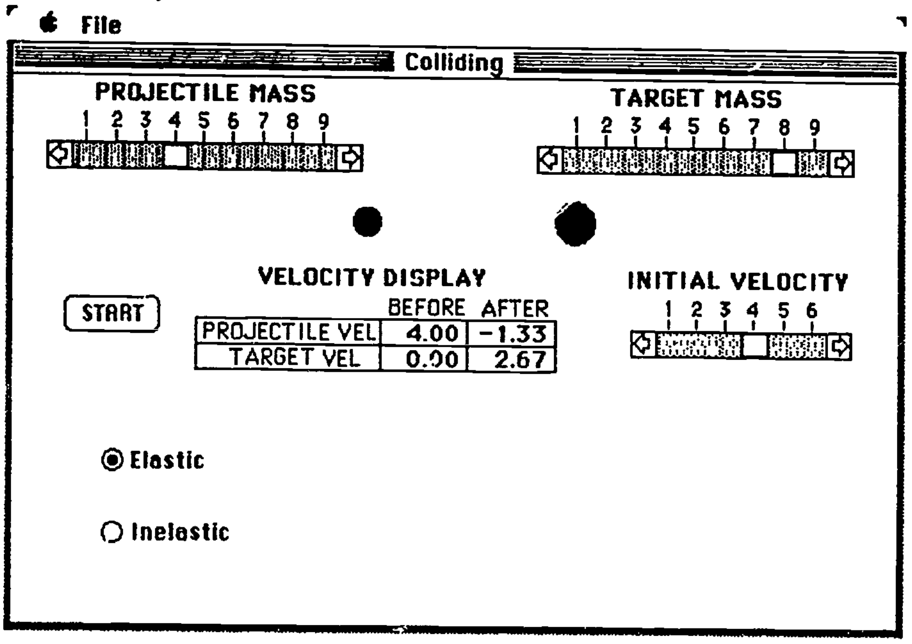
Work in progress

At the time of writing this paper eight application packages have been completed in preparation for trialling. It is anticipated by December that a substantially larger number will be available for demonstration, for assessment and for feedback suggestions. Screen dumps of representative portions of each of these are included for information.



	BEFORE	AFTER
TROLLEY VEL	4.00	

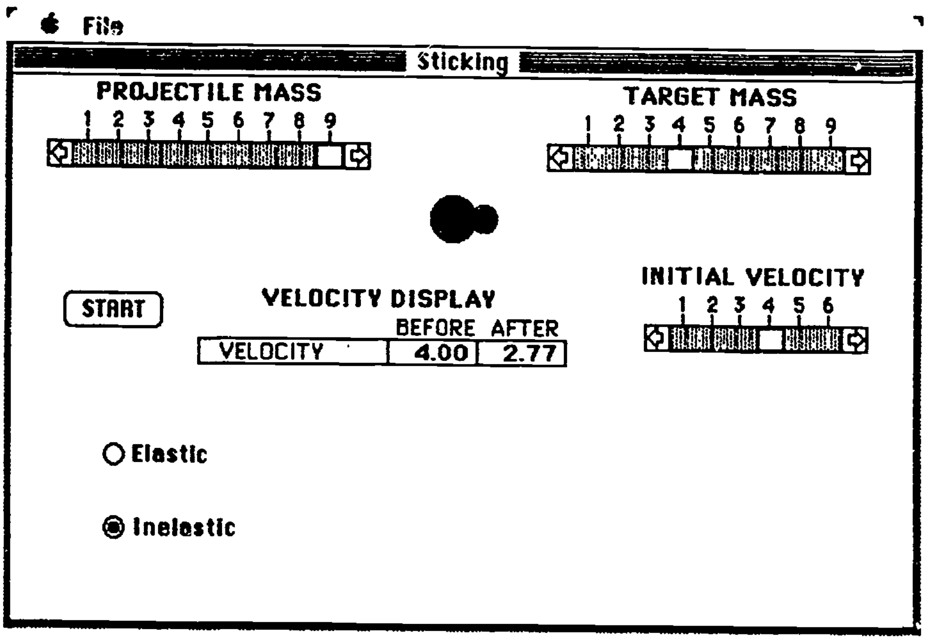
Figure 1. *Dropping*. Conservation of momentum is illustrated with student control over the mass of both trolley and brick.



	BEFORE	AFTER
PROJECTILE VEL	4.00	-1.33
TARGET VEL	0.00	2.67

[✓ Elastic]
Inelastic

Figure 2. *Colliding*. Elastic collisions are accessible for a range of masses from 1:9 to 9:1.

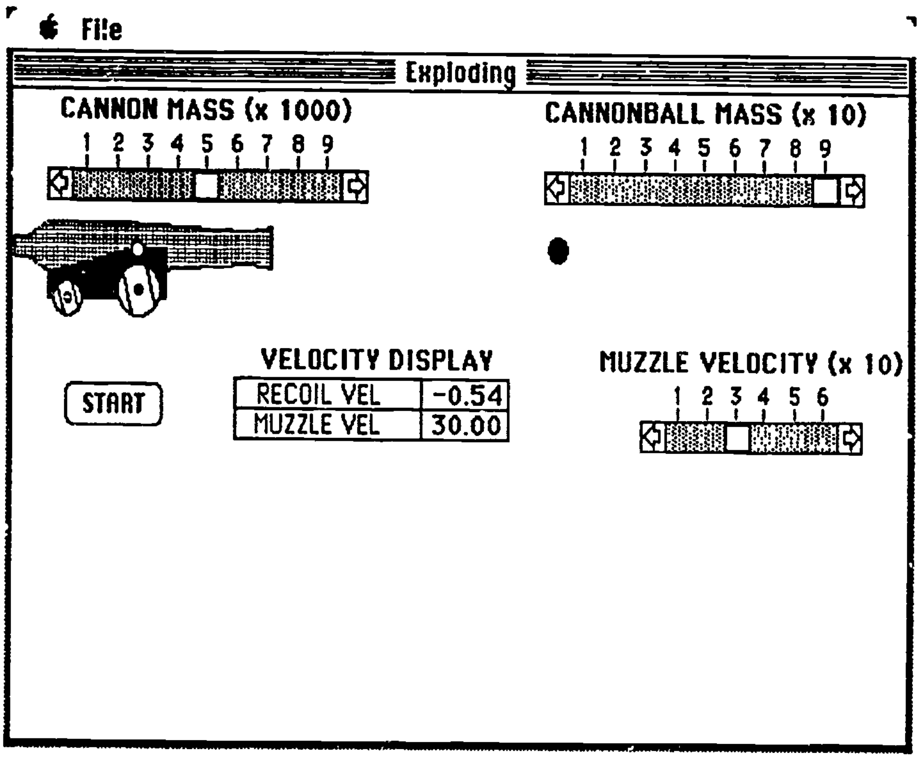


	BEFORE	AFTER
VELOCITY	4.00	2.77

○ Elastic

◉ Inelastic

Fig 3. *Sticking.* Inelastic collisions with variable mass ratios and total mass are available in this program.



RECOIL VEL	-0.54
MUZZLE VEL	30.00

Fig 4.1 *Exploding.* As with other momentum simulations this enables the student to vary the mass of the gun and projectile over a wide range.

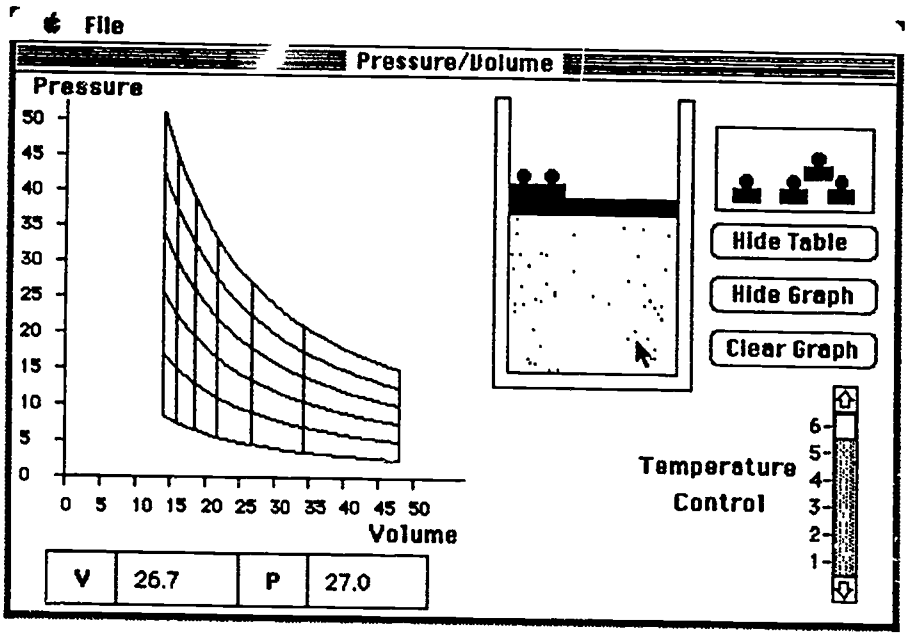


Fig 5. *Pressure/Volume.* The properties of a gas depend on a number of variables. Here a student can vary either the Pressure or the Temperature and observe the volume and pressure changes.

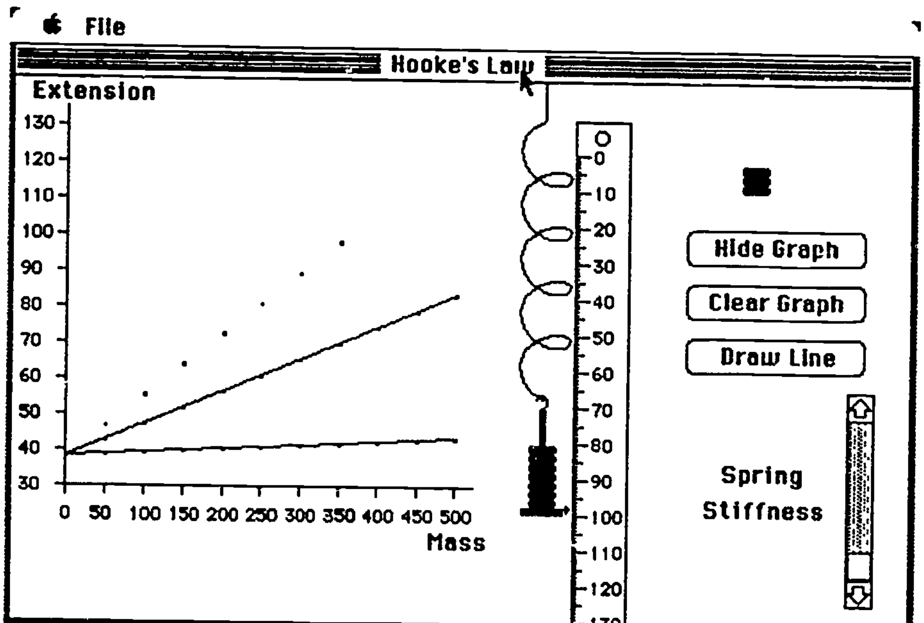


Fig 6. *Spring.* The extension of a spring of variable stiffness can be examined for a range of masses.

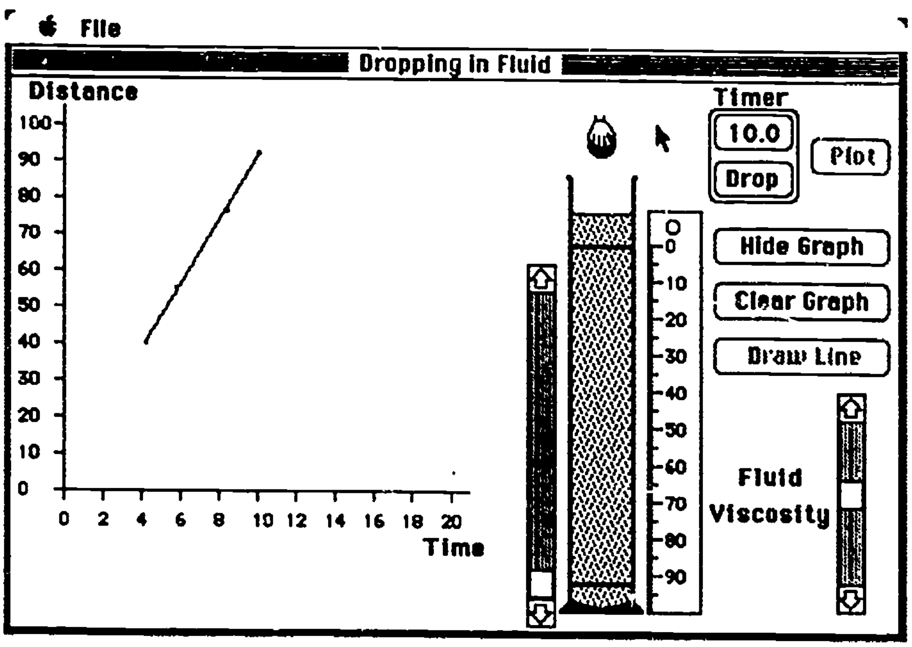


Fig 7. *Viscosity.* The terminal velocity of a sphere gives an excellent experiment for ratio and proportion.

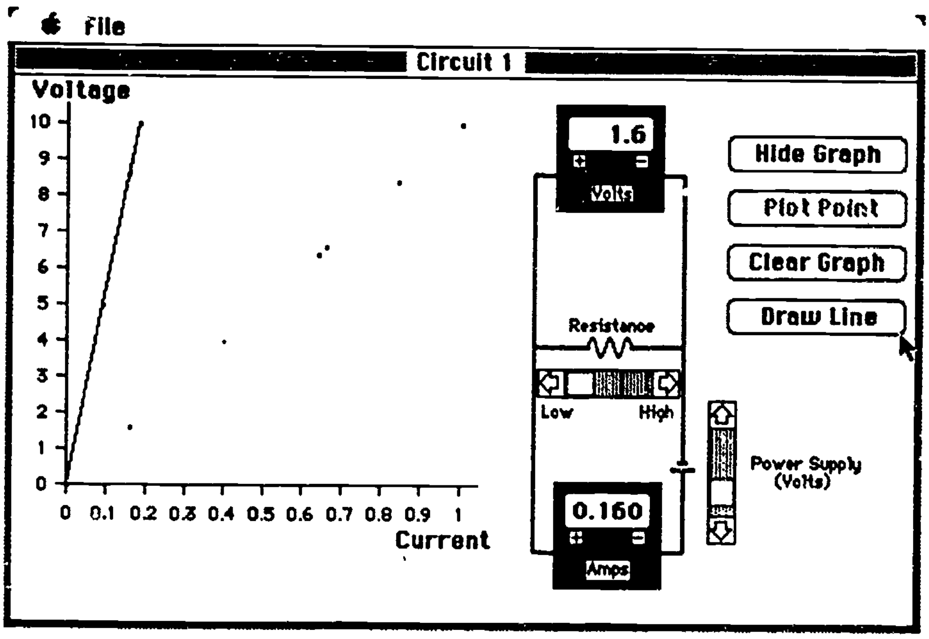


Fig 8. *Circuit 1.* The linear relationship between Voltage and Current can be seen as the student varies the control boxes and if needed the value of the resistance and the experiment repeated.

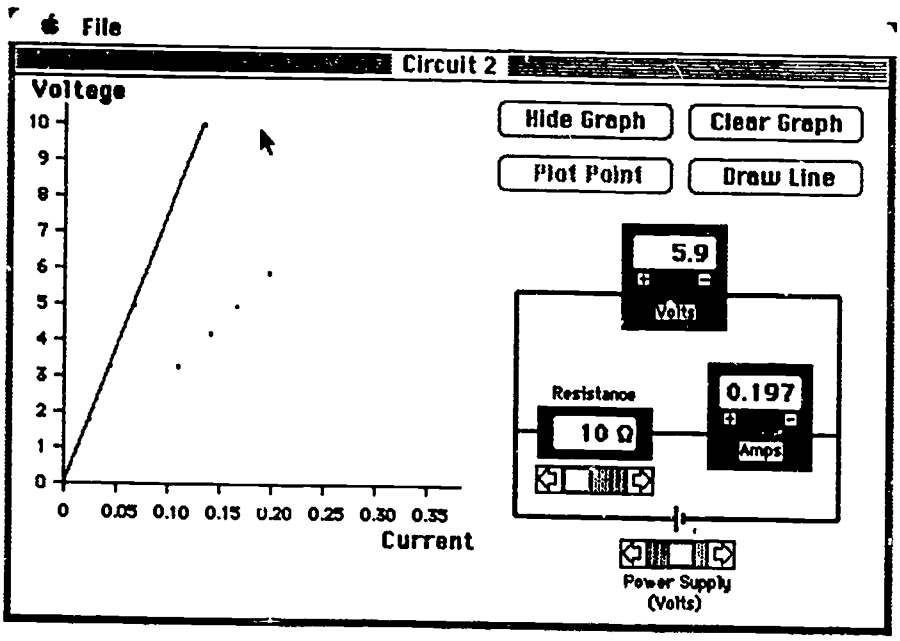


Fig 9. Circuit 2.

The original seven programs were to be in the areas of:

Motion
Forces
Momentum
Work and Energy
Properties of Matter
Fields
Experimental Methods.

The graphic sequences chosen sometimes fall into more than one of the above categories and this is done deliberately to enhance learning by familiarity. Thus Experimental Methods is concerned with ratio and proportion and will call on any sequence with linear characteristics as a suitable learner experience. The learner may also see the same graphic sequence elsewhere under Motion or Properties of Matter.

Conclusion

This project is far from concluded. It has opened up a wealth of opportunities and ideas for ways in which the teaching of Physics can be improved and specifically it has lead to our teaching staff having access to a number of instant experiments in a box (the Macintosh) for illustrating and supporting their teaching.

The sequences will be connected together and made available, on request to other institutions and hopefully will serve as a catalyst to foster further development and exchange of disks. One other institution has already indicated its interest in specifically developing teaching sequences and this could lead to further exchanges.

References

Bailey D.E., Logan P.F., Walker P.J., Hulme T., Drubetsky M. (1985). Physics Literacy Project, Student Control of learning, *Computers in Tertiary Education*, Bowden J & Lichtenstein S Eds.

Bork A. (1984) Computers and Information Technology as a Learning Aid. *Second Australian Computer Education Conference*, Sydney.

The Logichem Organic Inference Program

Frank R. Burden,
Chemistry Department
Monash University,

A program has been written, for the IBM/PC series, which deduces how a pair of organic chemicals would react with one another or with common reagents. It is a forward chaining inference system which makes use of separate knowledge base of rules concerning organic functional groups and multiple bonds. The knowledge may be added to or modified by the user, though the supplied set of rules is fairly comprehensive and covers the main reactions which students will encounter to the middle of their tertiary year. The system should be of use, therefore, to chemistry undergraduates as well as be of interest to others concerned with the representation of complex knowledge.

Chemical Computing.

Alongside the development of FORTRAN and LISP in the early sixties came the ever increasing use of computers in chemistry. In fact, in most universities the chemistry department was one of the main users of the computing facilities. Of course their work was all to do with number crunching and for the most part still is. Fortran is still the language used and will probably remain so for some time to come since it produces fast code and there is a large investment in the form of existing programs. These programs are available through an organized software sharing facility that has been in existence for over 15 years.

However chemistry as a whole is a mixture of science and art in the way that good cooking is, so that although recipes can be written down, it still takes a good chemical cook to turn the recipe into a product or invent a new one. So one way that I like to look at Organic chemistry is to compare it with a game of chess where the rules are definable (at least to some extent), the strategies are understandable but the game is is complex.

A student of organic chemistry has first to learn the language of valency and rules for the connection of atoms. Then the atoms are clustered together into so-called functional groups, which are groups of atoms that can be thought of as a whole and from which larger entities, molecules, can be built. There are no absolute rules as to what constitutes a functional group as against some arbitrary group but there is a general agreement with many exceptions.

The chemistry of a molecule can, in part be derived from its constituent functional groups and the bond types (single, double etc.) that connect them. This then forms a good start for thinking about the encoding of the rules of organic chemistry. Of course this is not the whole story else it would all be too simple to bother with; however it is enough for a beginning.

The original purpose of the program was to investigate the extent to which organic chemistry was amenable to being encoded in a logical manner. Somewhere along the way the aim was changed to producing a package that would be useful to students, after the scope and limitations of the original purpose had been defined.

The initial challenge was to encode the rules that relate to the reactivity of the functional groups and their associated bond types. The main criteria is that a pair of chemical compounds be typed in at the keyboard, or retrieved from a file, and a set of possible product molecules be displayed after computation. The chemical input and output to be as near as practicable to normal chemical conventions and the rules to cover the main possibilities.

The language choice falls between a procedural language such as Pascal and a declarative language such as Prolog. However one of the constraints in the present case was that the program should be understandable and modifiable by a range of chemists and chemical undergraduates. Pascal was chosen since it also has some natural advantages for chemistry in that its record and pointer types match the atomic and bond properties of atoms in molecules. Hence each record can include several pointers to other records and so, between them, form a graph which can represent a chemical structure.

Turbo Pascal©, together with some extender packages, was chosen on the grounds that a large number of these compilers were already in use and therefore users would have the potential to change any source code provided. In the event so far only compiled code has been circulated.

Pointers and Valency

The program inputs a string whose syntax is as close as possible to the accepted way of writing a chemical formula; the limitations being in the super- and sub-scripting which is commonly used. Of course chemists prefer to write two-dimensional diagrams to represent molecules. Parsing such diagrams would be an extremely complex task and well outside the scope of this project though it must be recognized as a limitation and a barrier to the acceptance of such programs by chemists.

In fact it is not always recognized by 'computer illiterate' chemists that there is a difficulty in this area as they have become familiar with specialized chemical word processors which enable them to place such diagrams in their documents. The enormous difference between going from a string to a diagram and from a diagram to a string is not obvious to them.

The string then has to be parsed into functional groups, bonds and other chemical syntactical symbols, all of which is made into a linked list of symbol groups. This list is then further parsed into a graph where each node is a record containing chemical information pertinent to the functional group and pointers to other atom records.

At this point is is easier if the molecular graph is thought of as a mobile (as used over children's cots) with the functional groups as the objects and the valency pointers as the strings. The object of the rules is to manipulate, add to and partly alter the mobiles.

The rules that will decide how two molecules react with one another are really rules of interaction between functional groups and functional groups or between functional groups and bonds. Hence a list of all such appropriate pairs can be made and then compared with an external database.

The encoding of rules and the Action types.

The external database was referenced to a well known organic chemistry textbook (Morrison and Boyd, 'Organic Chemistry) and at present contains about 150 rules. The reaction list is first pruned of the non-reacting pairs and the remaining rules acted upon at the discretion of the operator.

The rules and the functional groups are all encoded with a two character code for compactness and ease of reference.

One of the codes contained within each rule is an action code. This code specifies a particular Pascal procedure which will combine and rearrange the original (reactant) molecules in a particular manner pertinent to the reaction under consideration. The other codes contained within a rule specify how the functional groups of the original molecules are to modified upon reaction. Finally there are also codes to specify the conditions under which the reaction might take place (eg. heat,acid etc.).

The result of using the action procedure is to produce one or two new molecular graphs which represent the product molecules, and can themselves be re-manipulated and even reacted further with other molecules.

The program.

The program as a whole has been constructed to present in a professional manner, in contrast to many extant chemistry tutorial programs. An in-built editor is provided for typing in the molecules; help windows may be switched on and off; information windows pop down when some rules are fired, and the disk may be browsed for pre-recorded data.

The program may be viewed as a chemical calculator where molecules are input instead of numbers. It does not, however, address the problem of synthesis which is a problem of greater complexity, though an attempt at a restricted synthesis program may be made next year.

The program should prove useful to Australian first- and second-year chemistry undergraduates and their teachers. It may also prove interesting to those who wish to brush up their organic chemistry and even to those who are interested in the representation of complex knowledge.

Marketing and copyright.

A part of this years program is to be devoted to the marketing and copyright aspects of programs written by academics. The law, and some of the larger software houses, seems to want to treat software in a similar manner to books. If this is the case then there is no problem as academics have always written books and usually kept the royalties, which rarely amount to a fortune. It is important, and in keeping with the new political philosophy, that individuals be motivated to produce innovative ideas. Program writing is a difficult, lengthy and often tedious business for which there is the need for the prospect of a positive bonus. If the nation is to keep up with the trends in the ever changing world of computing and computer aided learning then such encouragement is needed. Of course, where grants are provided it up to those concerned to negotiate their own contracts which may or may not involve the ownership of copyright.

In my own case I have attempted to market the program myself under the name LogiChem and an organizational name of ChemSoft, which are registered business names. I took this course as there does not seem to be a suitable avenue for distributing chemical software to chemists in a commercial manner. The second advantage of doing this is that one is able to interact with the business community on an equal footing, which in itself promotes a better understanding of other peoples problems in both directions.

Dr. Frank Burden is Senior Lecturer in Chemistry at Monash University, his research interests are in the quantum-dynamics of small molecules and the production of expert systems for chemistry.

Grabbing knowledge graphically

Kay Fielden
School of Information Sciences and Engineering
Canberra College of Advanced Education

Iconic interfaces of machines like the Macintosh provide valuable and novel ways in which knowledge acquisition may be enhanced. A knowledge acquisition tool has been developed to design and create problem solving tasks by simple graphic interaction. In a controlled domain where objects are mapped directly onto icons for simulation, a clear mental model can be formed by the user. Exploiting the control structures of the iconic interface, that is, menus, buttons and super-imposed windows forces a distinct and unambiguous mental model of how the simulation is operated. Adopting a structured approach in designing the interface enhances knowledge acquisition.

While acquiring knowledge from experts has been regarded as one of the major bottle-necks for the construction of intelligent systems (Young, 1984), the underlying problem appears to be in knowledge representation. Once expertise has been acquired by a specialist, that knowledge appears to be stored in the specialist's mind in 'capsule' form. Acquiring detailed knowledge therefore becomes a difficult step, as the specialist often can no longer articulate these details. Traditional knowledge gathering techniques: form filling, questionnaires and interviews all require a translation process by a knowledge engineer in order to get the knowledge into the system. Aiello (1987) suggests that a rich variety of knowledge representations are required to allow selection of an appropriate representation.

The translation process required by the knowledge engineer in traditional systems can be side-stepped by the use of simple graphic interaction. The knowledge engineer requires a deep understanding of the system domain to select an appropriate representation.

Simulation of an object-oriented environment is an ideal domain for utilizing direct graphic interaction. This paper will discuss a knowledge acquisition tool designed for use within an authoring system environment.

Authoring Systems Environments

Application of graphic interaction to authoring systems environments enables specialist teachers to concentrate on the domain under consideration, with the computer system becoming 'invisible'.

Any domain which can be represented as a state/space relationship with constraints on the movement and relationship of objects to each and to the domain space is suitable for simulation and graphic interaction.

By adopting the same strategy described by Finzer & Gould (1984): 'only things that can be seen can be manipulated', specialist teachers can design their own simulated tasks and implement their own creative ideas, thus eliminating the need for programmers in the design of software. Not only does task creation in such an environment constitute a new medium for the presentation of information and concepts, it also provides new ways to gather knowledge both about the domain and the control structure for the domain and for the simulated system.

Knowledge Acquisition Tools

Although many researchers agree that interface design is domain dependent (Hollan, 1984, Aiello, 1987, Norman 1987) the availability of a rich variety of knowledge representations enables the selection of the most appropriate representation. While presentation of the domain under consideration need not necessarily be directly related the representational system, the more direct the mapping, the clearer the mental model. The picture on the screen should be close to the real world. In the world of simulated physical environments, direct mapping from the real world is possible via the use of direct manipulation in an object-based system. Movement of physical objects in the real world can be mapped directly on to the simulated system by direct manipulation (Figure 1(a).

However, control of the real world system is often inarticulated, details of control having become automatic for specialist users. To the knowledge engineer, gathering information about control of a task is difficult. The mental model is not clear. The problem becomes one of mapping a fuzzy model on to an explicit set of instructions in the simulated system (Figure 1(b)).

By recognizing that the simulated world is a different world by virtue of the control mechanisms, structuring the iconic interface to provide consistent control mechanisms eases the task of learning a new mental model for control. While the ideal situation is to make the machine 'invisible' (Norman, 1987, Hollan 1984), there is still control of the device.

Graphic Interaction

Graphic display has long been used as a powerful tool, particularly in simulation

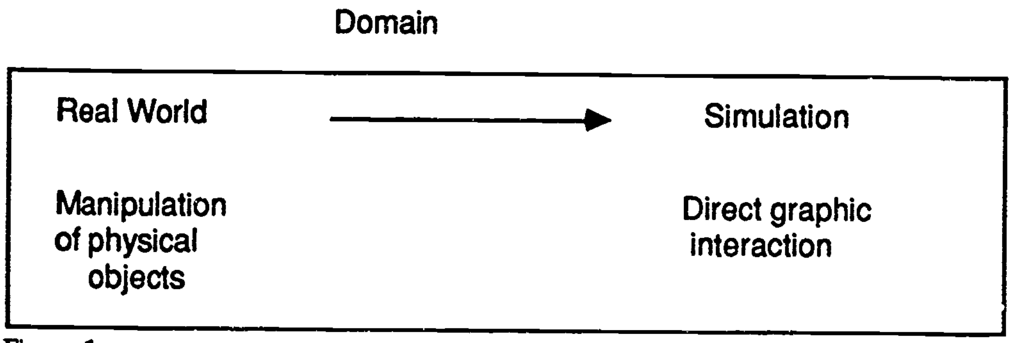


Figure 1a

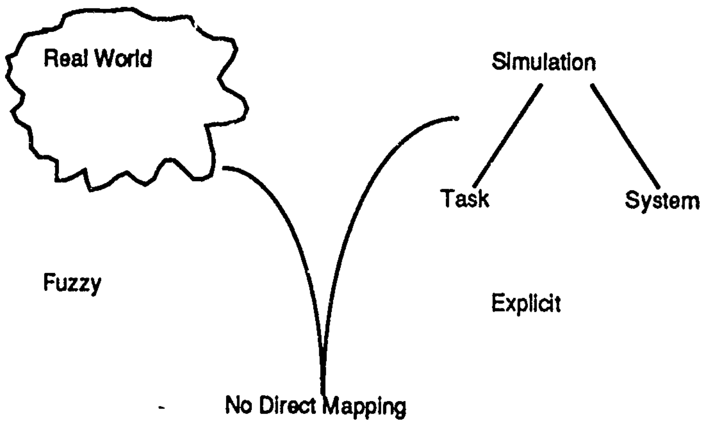


Figure 1b

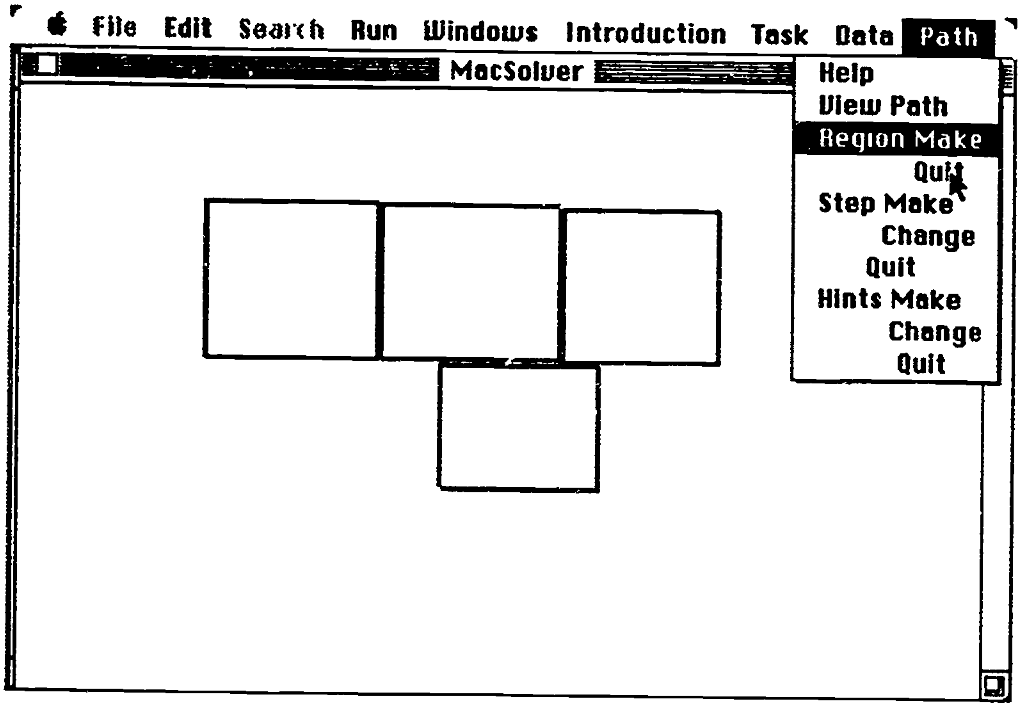


Figure 2

(Trollop & Alessi, 1985). Graphic acquisition of knowledge by direct manipulation has only become the subject of much research with the wider availability of machines capable of providing direct manipulation facilities. While it is difficult for other than a skilled artist to draw or paint pictures directly on to a screen using a device like a mouse, graphic input can be achieved easily and effectively in a variety of ways. If knowledge is required about rectangular regions on the screen this can be done by dragging a 'rubberband' rectangle. In an object-based domain, objects can be defined in both generic form and as instantiations of the generic form. Keeping a library of graphic objects in both these forms allows the user to request the appropriate object from this graphic library. It will be displayed as the instantiation if it is present, otherwise in the generic form. Direct manipulation of objects on the screen is then under mouse control, allowing the specialist teacher to build up steps in a solution path.

Control Structures

Icon interfaces like the Macintosh offer a variety of control mechanisms - pull down menus, buttons and superimposed windows, which can isolate foreign (to humans) control structures in a standard way, separating them from domain knowledge, thus differentiating the levels of activity required to interface with the simulated system.

Domain control must be made explicit in the simulated system. In the domain studied, teaching problem solving tasks in a restricted environment, the following strategy has been adopted:

1. control within tasks is by using pull down menus which are only visible when required (Figure 2);
2. display of a completed solution is by using buttons displayed on the screen (Figure 3).

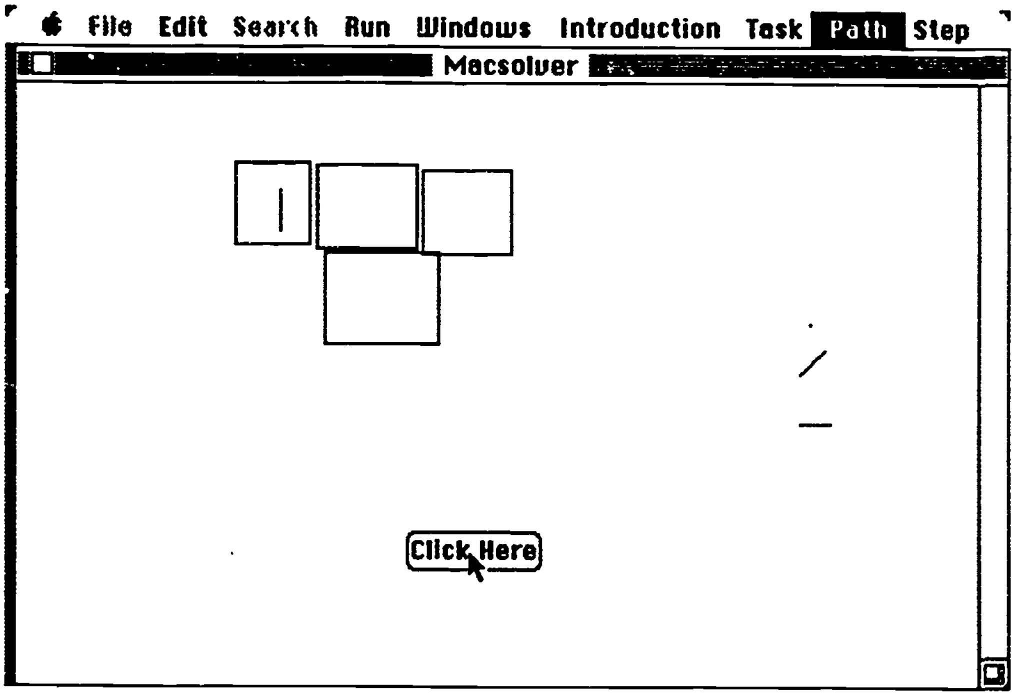
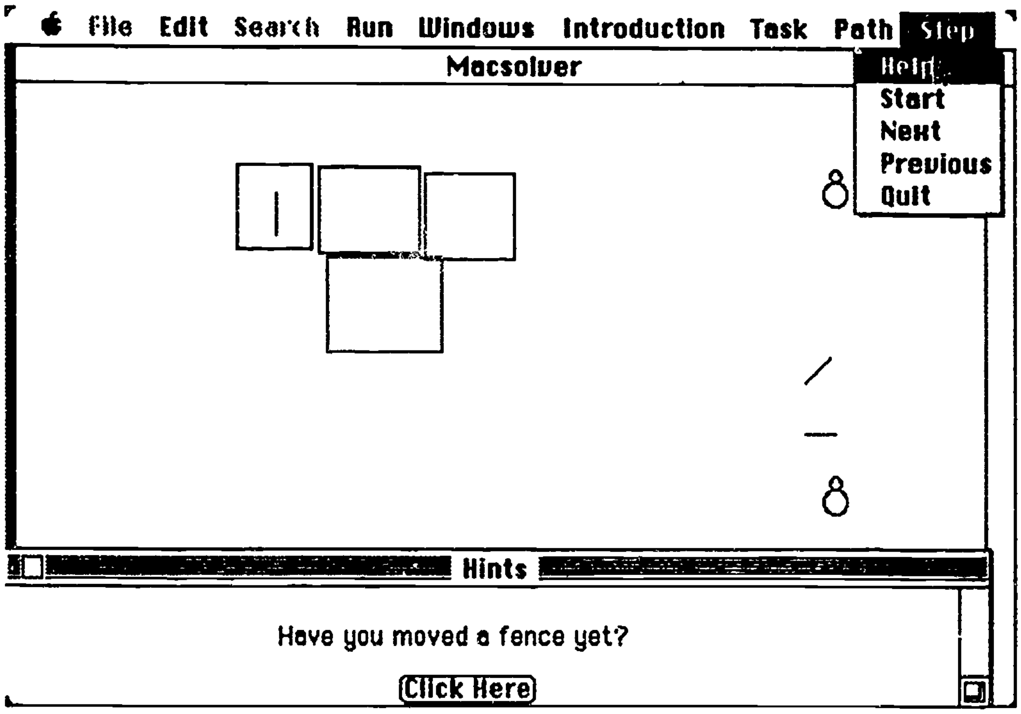


Figure 3



Have you moved a fence yet?

Figure 4

3. superimposed windows are used for displaying text for proving guidance to students attempting solution of a task (Figure 4).

Control is a function which exploits left brain activity and while menu selection is not sequential, it does provide a linear approach. Button selection gives access to a sequence of steps, and keeping graphic and text windows separate is consistent with the approach adopted for the Macintosh system interface.

There is at best a fuzzy mapping from the simulated world for control functions. This creates problems for the user who must form a mental model of control structures. However, separating them from domain knowledge by a completely different form of interaction with the computer system provides a clearly defined method of differentiating between control of the task and manipulating objects within the domain. By aligning these different levels of activity with a mapping most suited to left brain/right brain functioning (Edwards, 1979), the user, whether they be the specialist teacher at knowledge acquisition time, or the student at problem task execution time, can more easily form a mental model of the system.

Control features provided by the interface are available only when required. Pull down menus are only visible when a particular command is required and buttons and superimposed windows are sent away when they have been used. This provides minimum interruption to the thought patterns required to solve the task within the domain.

Conclusion

If the transformation from the specialists mind to the system is aided by an interface which is closer to the human specialists view of the problem than the system interpretation, it seems likely that the acquisition of knowledge both in quality and quantity, is enhanced. This knowledge acquisition tool attempts to shift the interface to the human end of the spectrum by making use of input techniques which reduce the 'visibility' of the machine. Hollan (1984) maintains that the design of an interface should be considered as the construction of a representational system for communication.

Further exploitation of the iconic interface provided by the Macintosh should lead to better understanding of the formation of mental models by different classes of users. Differentiation between control structures and domain activities at the Macintosh interface provide an effective means for improving formation of mental models. Adopting a structured approach to the use of an iconic interface for knowledge acquisition can also reduce the time taken to become at ease within the authoring environment.

References

Aiello, J.E. (1987). Why This Generation of Knowledge Representation Tools Will Fail. In G.Salvendy (Ed.) *2nd Int. Conf. on Human Computer Interaction*, Hawaii, Elsevier.

Alessi, S.M. & Trollop, S.R. (1985). *Computer - Based Instruction.* New Jersey, Prentice-Hall.

Amarel, S (1984). Expert Behaviour and Problem Representation. In A. Elithon & R. Banerji (Eds.), Ch.1, *Artificial and Human Intelligence*, (pp 1-41). Netherlands, North-Holland Publishers.

Benzon, B. (1985). The Visual Mind and the Macintosh. *Byte*, January 1985, 113-130

Edwards, B. (1979). *Drawing on the Right Side of the Brain.* Los Angeles, J.P. Torcher Inc.

Fielden, K. (1986). *Designing an Interface for Knowledge Acquisition for an Intelligent Tutoring System.* 17th Annual Australian CAE Computer Conference, Darwin, (pp 29-54)

Fielden, K. (1987). *An Enhanced Interface for Improved Knowledge Acquisition.* 2nd International Conference on Human-Computer Interaction, Hawaii, August 10-14.

Finzer, W & Gould, L. (1984). Programming by Rehearsal. *Byte*, June 1984, 187-209.

Hollan, J.D. (1984). Intelligent Object Based Graphical Interfaces. In G. Saluendy (Ed.) *Human Computer Interaction*, (pp 293-297). Netherlands, Elsevier Science Publications.

Mylopoulos, J. & Leuesque, H.J. (1984). An Overview of Knowledge Representation. In M.L. Brodie, M.J. Mylopoulos & J.W. Schmidt, J.W. (Eds.), *On Conceptual Modelling*, U.S.A., Springer-Verlag.

Norman, D. (1987). User Centered System Design: Emphasis Usability and Understanding. In *Proc 2nd International Conf. on Human Computer Interaction*, Hawaii.

Simon, H.A. (1983). Search and Reasoning in Problem Solving. In J. Pearl (Ed.), *Search and Heuristics*, (pp 7-29). Netherlands, North-Holland Publishers.

Young, R.M. (1984). Human Interface Aspects of Expert Systems. In *Expert Systems State of the Art Report*, (pp 101-111). Exeter Pergamon Infotech Ltd.

Kay Fielden is currently lecturing in Information Systems in the School of Information Sciences and Engineering at the Canberra College of Advanced Education. Research interests in CAI are both within the School as joint co-ordinator with Jo Baskett on a pilot project examining the most effective way of introducing CAI material into first year courses and also ongoing research into iconic interface design continuing from higher degree work already completed. The interface design project is funded by a CCAE research grant.

What do you say after they press "RETURN"? — Intelligent feedback in CAL

Geoff Foster
Tertiary Education Institute
University of Queensland.

Much of the value of Computer Aided Learning stems from its interactive nature, its ability to elicit from the learner a thoughtful response to a question; it is the very next step that is often under-exploited. The computer's reaction to the learner's response can contribute a great deal to the learning that is going on. If we simply tell the learner that the answer is right or wrong (or, at most, congratulate or commiserate), then we fail to take advantage of the receptive state that has been induced in the learner by the concentration needed to construct that answer. The repertoire of skills used by skilled tutors in discussion classes has its counterpart in CAL; maximum benefit can not be got until we exploit it. This, in CAL as in a tutorial, demands both a careful analysis of the student's answer and a reaction to it which has been developed to at least the same standard as the body of the lesson.

As in so many other disciplines, we workers in Computer Aided Learning are apt to fall into habitual ways of thinking and doing. It is constructive to do this to a controlled extent; progress would be slow indeed if we had to start off each new project from scratch and were not able to profit from experience, if we had always to 'reinvent the wheel'. But, from time to time, we ought to scrutinise some of our assumptions and practices, in case conditions have shifted without our noticing, making those assumptions and practices less valid (or even completely invalid); perhaps we might find that we no longer need that wheel, but would do better with legs or caterpillar tracks.The field that is sometimes referred to as 'Tutorial CAL' is a case in point; let us use it as an example. There is a number of possible definitions of what it is (and there is considerable looseness in their application), but for the purposes of argument I shall take that given in Higgins and Johns (1984), as a description of 'drill-and-practice' programs. It will be recognized as a paradigm followed by many CAL lesson authors (who would often, I am sure, vigorously deny that what they are doing is 'drilling').

Higgins and Johns list (page 39) the following components:

1. select task or question
2. display question
3. accept student's answer
4. match answer against acceptable answer(s)
5. report success or failure
6. adjust student's running score 7. return to (1).

This list needs to be only slightly modified to make it apply more generally. Specifically, if we widen items (1) and (2) to read:

1. select information, stimulus material and/or question
2. display information, stimulus material and/or question then it will cover a considerable number of those lessons which are commonly encountered.

In this paper, I shall focus particularly on items (4) and (5) and suggest that it is in these that the greatest potential for improvement resides. Sometimes, I suspect, attention has been almost entirely directed to devising stimulus material that will be searching and challenging, and displaying it in captivating ways, steps (1) and (2). I do not deny the value of these, merely recommend that steps (4) and especially (5) receive a comparable share of attention.

A mathmagenic approach

There is an extensive literature concerned with learning from texts, largely initiated during the heyday of programmed learning, but transcending it; the newcomer will find Jonassen (1982) a useful starting point. Among the many aspects of the field, one referred to as mathemagenics' has a particular relevance to CAL. The term properly refers to those behaviours in a learner (originally in one engaged in working through a programmed text) which 'give birth to learning', but has been given an extended currency as description of a domain of study in its own right.

The central notion of mathemagenics is that the structure and components of a text (or, by extension, of a CAL lesson) can be used to stimulate particular mental behaviours, particular ways in which the text is attended to and processed by the learner:

By controlling the way the learner thinks about textual material, we can determine how learners will convert presented information (nominal stimuli) into encoded information (effective stimuli). (Jonassen, 1982, vol 1, p 220)

The means we have of effecting this control lie mainly, but not entirely, in the content, structure and appearance of the textual material itself; Isaacs (1986) has written on the application of some of these techniques in CAL, particularly on the use of screen presentation to achieve advance organization and to demonstrate structure and emphasis. For printed text, some mathemagenic aids which have been used or proposed include:

- questions inserted in the text (either 'pre-questions' or 'post questions')
- advance organizers
- behavioural objectives
- headings, titles and emphasis (e.g. italics or underlining)
- instructions about note-taking (in the text, or separate). (Jonassen, 1982, vol 1, pp 130-131)

There is an obvious relationship between some of these text-based procedural devices and the parts of a tutorial CAL lesson that were listed before, and we should certainly bear these in mind; but when we are talking about computer mediated presentation rather than about print there is another very powerful mechanism at our disposal, thanks to the much more interactive nature of that medium compared with 'passive' text.

We should certainly use mathemagenic techniques borrowed directly from the world of print, but we should also take advantage of the receptive state of mind which these techniques induce in the learner. At the moment when he or she has just answered our carefully designed question, we can analyse that answer and then immediately respond to it, giving feedback to him or her which is sensitive to the answer and the context in which it was given. This will have more impact than a stock response can ever have.

So there are at least three phases of lesson design involved at each point of interaction with the learner:

- the construction (both in content and form) of the stimulus text, and of the task or question to be presented
- the analysis of the learner's response
- the feedback given as a result of that analysis.

I shall not dwell in this paper on the first of these, since there is little difficulty in seeing the relationship between established text techniques and their equivalents in CAL. However, there is not the same established body of knowledge and experience for the latter two items,which will therefore be addressed in some detail below.

Evaluating responses

A great deal of effort has been spent over the years in devising efficient and effective ways of evaluating the responses that a learner gives during a CAL lesson, which are usually (but not always) answers to questions. There is, as always, a trade-off; here it is between forced and free responses.

Forced responses, such as are used in multiple choice questions or for menus, are easy to analyse, but limit the learners' activities to those of choice or recall from a limited set of possibilities; free responses leave more to the learner, but are notoriously more difficult to evaluate.

The usual compromise solution is to allow a certain degree of latitude in response, and accept the fact that some answers which a human would judge acceptable will be rejected and some unacceptable ones accepted by the essentially limited algorithms that are practicable (in terms of programming complexity, processor time, memory or disk space, and so on).

But there is another underlying (and more important, to my view) aspect of the approach to evaluation which will bear some scrutiny. For both closed and open response sets, answer evaluation in the bulk of CAL lessons in service has been implemented in terms of 'matching', meaning that a learner's answer is compared with a set of stored expected patterns, some correct, some incorrect; this, indeed, was an explicit component of the drill-and-practice example quoted earlier. The learner's score and the subsequent course of the lesson depends on whether a match was made, and what it was. I prefer to think more broadly, of classification rather than of matching.

The central concept of matching is that of selecting a small number of predetermined patterns from a large population of possible ones, and subsequently winnowing out 'correct' from 'incorrect'. To be sure, this is a form of classification, but if we set out with the idea of classification as the primary one, it seems to me that we are at liberty to choose our categories in a freer, less encumbered way than mere matching will allow. If we start with the idea of matching, then we can implement certain restricted forms of classification; but if we start with classification, then matching is but one of the many options available to us (and may sometimes, indeed, be appropriate).

It would be, perhaps, a little fanciful to claim that what we are attempting here amounts to introducing a higher order cognitive skill to our programs (in the same sense that we think of analysis as a higher-order skill than recognition when we are discussing human learning); to do this to any extent would take us into the area of artificial intelligence, and I am not intrepid enough an explorer to venture as far myself; pioneering work is proceeding elsewhere, in several fascinating directions. However, even the move from matching to classification as our basic model will, I think, have a significant influence on the design of lessons, and even more importantly on the design of authoring tools (programs or languages designed to assist the 'courseware author') and, in the long run, on computer-based management systems.

As illustration; some present authoring systems stipulate that a certain set of consequences must invariably follow a 'correct' response, another set must follow an 'incorrect' one; no provision is made for any other case. Or, in a more specific instance, some packages insist that a learner be given all the marks for a right answer and none of them for a wrong one, thus effectively ruling out answers of intermediate merit. These sorts of restrictions can be frustrating to someone trying to write lessons intended to exercise learners' higher-order skills; they exert a similarly damaging but subtle influence on the field of CAL as a whole.

Perhaps, in CAL, it would be better to replace the right/wrong duality by a continuous spectrum which ranges from 'excellent' down to 'awful', even if this does have to be implemented in practice as a set of disjoint categories. I still remember being told, thirty years ago when an engineering student, that in engineering there are no right and wrong answers, just good and bad ones; this is true of more fields than engineering.

We have been talking of 'tutorial' CAL; a term which, presumably, is intended to suggest that there is some relationship between the activities in the CAL lesson and what normally goes on in a conventional class conducted by a tutor. One of the desirable characteristics of a human tutor (as well as those of being knowledgeable and skilful) is that he or she listens to his or her students and responds sensitively; it is this behaviour that we ought to be trying to simulate here. Just as I am accustomed in my work as a staff developer to running workshops intended to enhance the listening and communicating skills of tutors, I would like in this paper to advocate that we enhance the corresponding 'skills' of CAL lessons.

A tutor who is listening with sensitivity can do much more than determine which answers are right and which wrong; he or she can detect subtleties and, with experience, diagnose mistakes in reasoning and deficits in prior knowledge. There ought to be similar benefits in the refinement of classification that we are discussing here.

Once a approach based on classification has been adopted, there are several tasks to be undertaken at each point in the lesson at which the learner gives a response:

- constructing a set of meaningful categories and specifying program paths or reactions to each of these
- deciding on the criteria by which responses will be sorted into those categories
- designing algorithms by which the criteria will be tested
- checking the algorithms against test data; responses generated either by the lesson author or, preferably, by a pilot group of learners who have similar characteristics to the 'target audience'
- developing categories, paths, criteria and algorithms in an iterative way.

The criteria for classification mentioned here, and the algorithms by which they are implemented, should be developed to an appropriate level of sophistication, where 'appropriate' is the key word. It is just as ineffective to waste resources and effort in using refined techniques to make gross decisions as it is frustrating to try to make subtle discriminations with too coarse a sieve.

(Note also that the development process referred to in the last item in this list is not a substitute for that which takes place (one would hope) during the production of any new learning material, but is contained within it.)

It was not by chance that I included two distinct activities in the first item in the list.

The processes of deciding on categories and on the differential behaviour of the program (lesson) in respect of these categories are intimately linked. One might use here an analogy borrowed from medicine; a medical researcher who wishes to determine the aetiology of a patient's condition may use different tests from that of the clinician who wishes to make a diagnosis for the purposes of planning treatment. In the same way, we must be clear to make the distinction between those questions which are appropriate for assessing a learner's state of knowledge and ability and those which will enable us as teachers to best facilitate his or her learning. I am not saying that there is never an overlap between these two sets.

Intelligent feedback

I have made the point in an earlier paper (Foster, 1986) that there are occasions on which we may wish not to respond to a learner's answer at all and others on which a brief response is all that is warranted. I claim here, however, that as a general rule, if we fail to exploit the opportunity of responding fully, then we are missing a chance to enhance significantly the very learning that we are trying to induce. After all, the learner has just been deeply involved in constructing the best answer that he or she can, and his or her attention is likely to be tuned to a fine pitch. We are poised at the legendary 'psychological moment'.

A well designed response, moreover, should act as a powerful motivational reinforcer; if the learners think they are being 'taken seriously' they are less likely to be turned off than if they gradually get the feeling that, however hard they try, the computer response is going to be a laconic "correct" (or even worse a patronising "Well done, Johnny!").

At the very least, the response should acknowledge the learner's answer by echoing it (or better, paraphrasing it) so that he or she can see that 'the computer has heard and understood it'. After that, the learner could be given reinforcing or correcting information, which could in its most elementary form be the words "right" or "wrong", but would be better if elaborated appropriately. This might be followed by new information, designed to consolidate or expand on the textual material which preceded the question just answered.

The point at which the 'response' ends and the next part of the program or lesson begins is somewhat undefined; we could choose to say that all material which is sensitive to the learner's most recent answer is part of the response, but this would only be the case when the subsequent route through the lesson is fixed; ideally this would also depend on the learner's 'answering history'. In any case, it is more important to include all the features here described than to attribute them to particular functions of the underlying software.

I have said in the preceding section that it should be our aim to emulate the sensitive and knowledgeable tutor in the behaviour of our programs when they respond to a student's answer. One of the components of sensitivity, as I have mentioned, is the ability to listen; the CAL equivalent of this resides in the operation of the procedures used for evaluating, for classifying responses into categories. The CAL analogue of knowledgeability in a tutor is embodied in the prior definition of the categories and in the design of the algorithms which perform the classification (here again it is difficult, and of little point, to distinguish absolutely between these two).

Another component of sensitivity is empathy; the nearest equivalent of this in a CAL system is in the extent to which the program as a whole is designed to suit the target audience and, on a more local scale, the extent to which it can respond to what it 'knows' about the learner; the history of previous answers and interactions earlier in the session and, maybe, in other sessions. As a concrete example: if a student has already demonstrated his or her knowledge of a fact, we would not consider it tactful if a tutor were to carefully explain the fact to him or her at a later time; in a like manner this should be an elementary requirement of all CAL software.

Remarks on implementation

I have recommended a move away from matching algorithms toward something more general. This was not simply an expression of faith; there have been developments in computing going on outside the CAL field, brought about by the revolution in computer use in office practice, that it would be remiss of us to ignore. (As well, as I have mentioned, as some valuable research aimed specifically at CAL; I am sure we shall see evidence of this work elsewhere in these proceedings.)

It is commonplace, now, to find on-line spelling checkers and thesauruses ('thesauri'?) available as accessories for word processing packages used on personal computers in offices. The thesaurus, in particular, offers features very close to what would be needed for intelligent classification of learner responses; and both these types of auxiliary programs have been developed to the point at which it is possible for them to process each word as it is typed in by the user in 'real time', without serious delays (especially as learners will be unlikely to be typing as fast as clerical staff).

What the thesaurus does now is very close to what would be required of answer classification algorithms; it is capable of taking a target word and associating it with a list of synonyms, close relatives and antonyms. The same type of coding techniques can equally be used to associate an answer with the appropriate one of a number of response categories predefined by the lesson author. Further; just as in an office the thesaurus to be used on any given occasion, or the dictionary used for spelling checks, are selected from a small number of standard versions, so the same generalization can be applied to answer evaluation. One might envisage having an answer thesaurus for psychology and a different one for political science; certainly it would be impracticable to construct a new one for every new course, and ludicrous to do it for each new lesson.

The other technique that has been developing quietly outside the CAL world is that of so-called 'natural language interfaces', mainly for use with database packages. Just as with thesauruses and spelling checkers, extremely keen competition has stimulated advances in this field that are equally worthy of consideration by CAL workers. Why should the lesson author be compelled to construct complex logical expressions (('SPAIN' OR 'PORTUGAL') AND NOT ('ITALY' OR 'FRANCE')), while his or her colleague in business is benefiting from much more sophisticated and easily used techniques?

When we turn to the question of providing feedback to the learner, there is no similar advance in computational technique that springs to mind; the requirements here are very similar to those needed for presentation of the primary information in the lesson. So, as in that case, we should simply make sure that we are not overlooking any ways of making the content and presentation of the text as effective as possible; here we are using the corpus of expertise of the technology of text directly. However, it is worthwhile, even at this stage, asking the fundamental question which I raised in Foster (1984); should we be using the computer exclusively to present this information, or should we also rely on supplemental material, even on print material, using the computer to manage this information rather than to present it?

Conclusion

What I have attempted to do in this paper is to make two major points; the first is that Computer Aided Learning has a distinctive strength, compared with some other media of teaching and learning, in that it is interactive, and therefore uniquely capable of demanding thinking and reasoning from the learner. I followed this with some suggestions as to how best to exploit a particular aspect of this quality, which we could sum up in the aphorism "strike while the iron is hot".

The second point made and elaborated is that we might try to emulate that quality of effectiv: tutors which we refer to as sensitivity, and add it to the equivalents of knowledge and skill that, presumably, we have always tried to incorporate into our CAL lessons. This might be done, on the one hand, by paying attention to ways of evaluating student answers that go beyond mere matching or identification with predefined models; on the other by putting as much effort and ingenuity into constructing truly helpful feedback to students as we have always done in devising rich text and searching questions.

Implicit in these suggestions is a further technical one; that we should not fail to take advantage of computational techniques developed for other purposes; such as the on-line spelling checker, the thesaurus and the natural language interface.

There are many exciting developments which are only just over the horizon of the CAL world; my proposals here may be more modest in scope than some which will be considered commonplace in the future, but I hope they have their place and that they might offer a lasting benefit.

References

Foster, Geoff. (1984). The Computer-Managed Book. In R. M. Russell (ed), *Proceedings, 2nd. Annual Computer Aided Learning in Tertiary Education Conference*, Brisbane, University of Queensland.

Foster, Geoff. (1986). The Case for Plain Text CAL. In G. Bishop & W. van Lint (eds), *Proceedings, 4th. Annual Computer Aided Learning in Tertiary Education Conference, (CALITE 86)*, Adelaide, University of Adelaide.

Higgins, John and Johns, Tim. (1984). *Computers in Language Learning*. London, Collins Educational.

Isaacs, Geoff. (1986). Screen Design for Computer Assisted Learning. In G. Bishop & W. van Lint (eds), *Proceedings, 4th. Annual Computer Aided Learning in Tertiary Education Conference, (CALITE 86)*.

Jonassen, David H. (ed.). (1982). *The Technology of Text*, (Volumes 1 and 2), Englewood Cliffs, Educational Technology Publications.

Geoff Foster is a lecturer in the Tertiary Education Institute (TEDI) at the University of Queensland. He began his academic career as a lecturer in mechanical engineering, but his interests gradually shifted away from engineering and towards teaching and learning. He made the move to TEDI in 1974 after fourteen years of teaching, and since then has been engaged principally in general consultation work on aspects of teaching and learning with academic staff. His present interests include CAL but are not limited to that field.

Some lazy thoughts on applying AI to CAL

Marshall Harris
Department of Computer Science
University of Queensland

The leading edge of AI research in CAL is mainly concerned with problems of representing knowledge, modelling students' knowledge, processing natural language, and intelligent choice of teaching strategies. It will be some years before this research becomes economically and educationally viable.

Meanwhile, there are many aspects of CAL that can benefit from piecemeal application of AI techniques, without attempting to resolve the above-mentioned difficult problems. The author is currently implementing a system where AI techniques are applied to a conventional CAL system, to select suitable material for delivery to the student. This was described in (Harris, 1985). This paper considers another area where such techniques could be applied, namely Authoring Systems.

The Lazy Person's Technique.

In (Harris 1985) I suggested that a considerable time would elapse before a full, "intelligent" CAL system would be truly user-friendly, affordable, and acceptable to teachers and students as a viable teaching/ tutoring alternative to human teaching.

Meanwhile, however, it is clear that techniques imported from AI systems can be applied in a somewhat piecemeal fashion to various aspects of a CAL system, with useful and effective results. Such systems are exemplified in (Webb, 1984,1985) and (Harris, 1985).

Students learning computer science are taught the technique of Top-Down Decomposition of problems. This means, simply, breaking a problem down into ever smaller problems, until the level of decomposition is such that each sub-problem is easy to handle.

Another aspect of this technique is that it allows one to consider the problem at a high and very general level first, before considering the details.

Putting this another way, Top-Down Decomposition allows one to dodge the nasty bits and concentrate on the interesting parts. One might, indeed, say that it encourages one to put off till tomorrow what one does not feel like doing today! That is why I call it a lazy person's technique!

I mention this because I believe that whatever comprises CAL can be broken down into a bundle of sub-tasks or sub-problems, to each of which some AI techniques may be applicable and effective.

By identifying those sub-tasks where AI can be used, and deferring attempts to apply AI to those tasks which are currently intractable, we are likely to make more rapid progress in CAL than would be the case if we were to try to tackle the whole problem headon. Among the intractable tasks I include: understanding natural language, in speech and writing; generating a complete lesson from a subject domain knowledge base; effectively representing a student's knowledge of the domain; and other difficult tasks like these.

This is not to say that the "headon" approach should not be attempted, nor that the difficult tasks should be neglected: these are precisely the areas where well-funded research needs to done. But, while others tackle the hard parts, educators are anxious to "get on with it". So let's help them by trying to improve the CAL techniques we now have, in those areas where improvement, via AI, is currently feasible. It was in this spirit that I proposed the PSICAT system (Harris, 1985), now under development, where AI techniques were applied to the problem of selecting conventional CAL material appropriate to the student's current progress record.

In this paper I want to consider yet another sub-task of CAL where I believe AI could usefully be applied, namely, Authoring Systems.

Achieving flexibility in an Authoring System.

In (Harris, 1983) I described the features which I then believed an Authoring System should contain. That paper was written before I fully realised that, for CAL to progress, AI must somehow be involved. Certainly the kind of flexibility and the facilities I advocated then, are still highly desirable. However, I hope to show here that much of that flexibility can be provided automatically by the built-in expertise of an AI-based Authoring System (AIBAS). In order to show this, it is first necessary to analyse and decompose the necessary features of a good Authoring System.

What an Authoring System should be like.

There are essentially three kinds of people who whose interes are affected by an Authoring System:

- the teacher, in the role of educator;
- the teacher, in the role of lesson author;
- the student, in the role of learner.

I shall refer to these roles as "educator", "author", and "student" respectively.

The needs of the educator: teaching style.

The educator requires pedagogic flexibility. By this, I mean that the lesson, or rather, the Authoring System that is used to produce the lesson, should support many pedagogical styles and a mixture of styles.

Styles would include "browsing" - an unstructured presentation of material giving the student free choice in the selection of material; structured or hierarchical organisation of material; and others.

A fully intelligent system would include a knowledge base of criteria which would enable it to decide upon which pedagogical style or styles it should use for a given lesson or lesson segment presentation.

In (Harris, 1985) I have described a system (PSICAT since renamed "ICATS") which presents a lesson composed of hierarchically organised lesson-segments, selected according to the student's performance record. I shall not discuss this particular aspect of AIBAS further in this paper. In the light of my opening remarks, I also do not intend to pursue the problem of establishing a system that can make its own decisions about style.

Rather, I would propose a system that can advise the educator on the appropriate style of presentation, leaving the final choice to the educator. This would be, in other words, an advisory Expert System - a sort of "Educator's Style Adviser". This puts the intelligent Authoring System well within the capabilities of currently available Expert Systems.

So the Educator's Style Adviser would interrogate the educator to find out what s/he has in mind for the lesson, in terms of the nature and structure of the content, and provide advice accordingly.

The needs of the author - hardware.

The distinction between the needs of the author and of the educator is somewhat artificial, but useful for establishing further areas where AI can be effectively applied.

The Authoring System must be capable of generating different kinds of presentation - text, graphics, colour, sound, simulation, and animation, and mixtures thereof; it must be able to control external devices such as video disc and tape machines, tape recorders, and slide projectors; and it must be capable of activating and interacting with other programs available on the computer, such as language compilers and command shells.

Seen from this point of view, an intelligent Authoring System needs to have a knowledge base with extensive information about the characteristics of various hardware devices and how to use them.

Assume that the result of the consultation with the Educator's Style Adviser is that the author has chosen the appropriate style, and that the Educator's Style Adviser has used this information to select another Expert System that can deal with the particular choices made. I shall call this system the "Author's Assistant".

The author now has to embark upon the task of creating the actual lesson, or lesson segments.

Authoring a lesson.

The Author's Assistant would be a single, self contained Expert System whose knowledge components (rules, facts, etc) are marked to indicate whether they are appropriate for the choices made at the Educator's Style Adviser stage. Only items which are appropriate are used by the Author's Assistant.

The Author's Assistant would be concerned with such things as screen layout, editing of text, graphics, and sound, the control and interaction of external devices, the nature of interaction between student and segment, and the manner in which student input should be evaluated.

Handling text screens.

Since the Author's Assistant is to behave as an expert, it should interact with the author at a high level, rather than require the author to provide detailed layouts, etc.

For example, the Authoring System would require the author to indicate what should be shown on the screen; say: text, a question, and a response. I would not expect the system to accept natural language input from the author: that is one of those areas of AI which I want to place, for a while, in the "too hard basket", and leave to the leading edge AI researchers! I would be content with a menu-based dialogue!

The Authoring System would then call for more information: i.e it would ask the author to enter the text, and automatically invoke a screen-based text editor (with plenty of on-screen help!) for the input. When the input is complete, the Authoring System knows exactly how much text there is. At this stage the Authoring System might ask the author to use the cursor to indicate where words should be highlighted or coloured. It then calls for the question, using the same text editor, and the style of question such as multiple choice, or text string (if not already established by the Educator's Style Adviser) and displays a provisional screen layout as it would be seen by the student.

A weakness with many non-AI Authoring Systems, and a difficulty for the author, is that the s/he has to supply pattern strings for matching against student answers. An AIBAS should generate such strings automatically. The system might use a combination of methods: "soundex", dictionary look-up (suitable techniques might be gleaned from currently available spelling checkers), transposition of characters, and so on.

Grammatical analysis of student input is more difficult for an AIBAS in the last analysis it is a natural language understanding problem, to be lazily filed away in the aforementioned too-hard basket! However, if we wish to deal with a restricted subset of natural language, or with a well defined programming language, for example, the Authoring System should provide facilities for syntactic definition.

Perhaps the Authoring System could provide two facilities: a means of defining a "complete" definition for the language in question, and a means of defining a syntax associated only with a particular answer. The latter seems more attractive for the lazy Authoring System, and is probably easier for the author to handle: I would expect the Authoring System to ask for lists of words, their associated grammatical categories, and a syntax description limited to the allowable structure of the answer currently being processed. This, combined with the techniques for pattern matching suggested in the previous paragraph, would enable the Authoring System to generate all the testing needed to evaluate textual student responses.

Handling graphics

Designing aesthetic and legible graphics is an expert job. It should therefore be done by an Expert System. It is, perhaps, again a too hard task to develop such a system.

On the other hand, the lazy AI-er should apply her/ himself to the design of a Graphics Editor with an intelligent ability to create graphics from the author's replies to questions. To avoid excessive complexity, the Graphics Editor should develop a display by a process of trial and error, refining its questions to the author and the graphics generated in response, until the author is satisfied with the result.

Handling other devices.

I do not profess to know a great deal about videos, tape drives, and the like, so I do not want to comment on this aspect in any detail.

Let it suffice to say that the principles on which the preceding discussion is based can be applied here too, with Editors and Expert Systems providing advice and, if necessary, generating appropriate procedures.

Some aspects of software design.

The Expert Systems proposed here would need, in their knowledge bases, not only rules and facts, but also procedures. These would be very high-level procedures which the Expert system would condition or parameterise, to produce the demonstration layouts, text screens, response evaluations, graphics etc.

These same procedures would, of course, form part of the final AIBAS system that interacts with the student.

The needs of the student.

It is a general requirement of CAL, whether AI-based or not, that system must respond quickly, provide help, perhaps allow user friendly consultation of databases, be generally easy to use and easy to see.

Other desirable qualities include some, at least, freedom of choice (which may, perhaps, be an option specified by the author) of material, ability to terminate a session and resume from the last point of exit, and, especially, adaptation to the particular student. This latter aspect is addressed in (Harris, 1985).

Conclusion: AIBAS is possible now.

By now it must, I hope, be clear that existing and proven AI techniques, especially Expert Systems, not only can, but must be used in CAL Authoring Systems.

It should also be clear that there is an order of priority of applica'ion of Expert Systems techniques, starting with the easiest application first. That order is, I believe:

1. the Author's Assistant;

2. the Graphics Editor;

3. the Educator's Style Adviser;

4. the rest!

The above order is probably the complete inverse of that which the leading edge AI researchers would assert. The point is that my suggestions are feasible and would be commercially and educationally viable within the current state of the art.

References

Harris,M (1983). *What an Authoring System Should be Like.* Proceedings, 1st CALITE Conference, University of Queensland, Brisbane.

Harris,M (1985). *PSICAT: a Pseudo-Intelligent Computer-Assisted Tutorial System.* Proceedings, 3rd CALITE Conference,University of Melbourne, Melbourne.

Webb,G (1984) *A Methcdology for Intermediate Level Knowledge Representation in CAL.* Proceedings, 2nd CALITE Conference, University of Queensland, Brisbane.

Webb,G (1985). *Student control under the feature network based courseware design methodology.* Proceedings, 3rd CALITE Conference, University of Melbourne, Melbourne.

Marshall Harris has been a programmer since 1959, and a Lecturer in Computer Science since 1971. Prior to that he worked, at different times, within the computer industry in both software design and applications programming. In 1983 he convened and organised the first CALITE Conference, at the University of Queensland. He has designed a structured CAL authoring language, which he used to produce CAL material for 1st-year Computer Science students. His current research interest is in the field of Artificial Intelligence applied to CAL, with a few sidelines in other applications of AI. He is currently implementing his Intelligent Computer-Assisted Tutorial system ("PSICAT" - see this paper) on a variety of machines.

Teaching Introductory Programming using a language-intelligent programming environment

Michael Newby, School of Computing and Quantitative Studies
Curtin University of Technology

The teaching of introductory programming normally involves teaching not only an approach to problem solving and the language itself but also an editor and some operating system commands. Learning to use the editor and the operating system has little to do with learning how to program but often causes problems because of lack of keyboard skills. Language-intelligent development tools are available to aid program coding and this paper describes the author's experience using the programming environment ALICE to teach Pascal to first year undergraduates.

When teaching introductory programming, no matter which language is being used as the vehicle, there are a number of factors apart from language that must be taken into account. Firstly, there is the method used to describe the algorithms, secondly the computer and therefore the operating system under which programs are to be run, and finally, the editor which is to be used to create programs. The program development process using the above approach is shown in Figure 1.

One problem encountered by first time users is slow input due to lack of keyboard skills, and in addition to this there is the need to switch between the various utilities, invoking them using operating system commands.

Becoming competent at using these utilities and gaining keyboard skills may well be valuable in themselves, but certainly have nothing to do with developing problem solving and programming abilities.

Various systems have been developed aimed at integrating such utilities. These systems include BASIC, UCSD Pascal and Turbo Pascal. All of them provide an operating environment which is menu driven and allows access to the file system. In each case, the editor is easy to use and switching between the utilities causes few difficulties. BASIC systems normally contain an interpreter and syntax errors are detected at run time and in the case of BASICA on an IBM-PC, the line containing the error is displayed allowing it to be corrected immediately. Both UCSD Pascal and Turbo Pascal contain a compiler and syntax errors are detected by it. In both cases, transferring to the editor is almost automatic and the cursor will be on the character where the syntax error was detected. Systems such as these have assisted learning to program, but in all of them the programs have to be entered fully every time and good keyboard skills are still necessary.

The programming environment ALICE (Looking Glass Software, 1986) was developed to remove the drudgery of coding programs in Pascal, by providing a language intelligent editor within a full operating environment. First year Information Processing students in this University were introduced to programming in Pascal using ALICE, and the remainder of this paper evaluates its use.

Overview of ALICE

The ALICE system is available for IBM PC and compatibles, running under the oper-

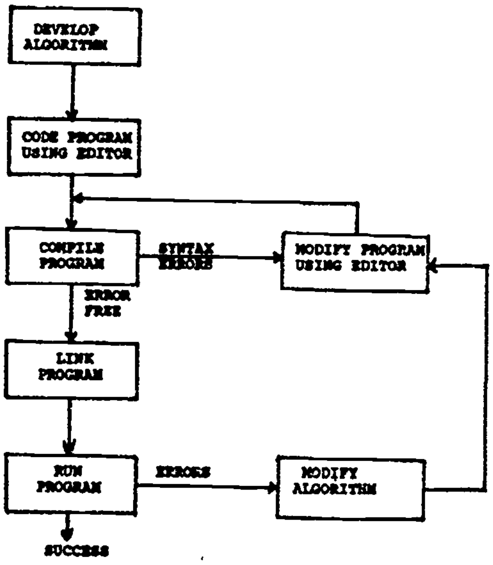


Figure 1

ating system MS-DOS Version 2.0 or higher and it consists of four components:

- the editor
- the interpreter
- the debugger
- the help facility

The editor is syntax-directed and designed specifically to create Pascal programs. This is done by providing templates for all program constructs. For example, a while loop consists of the template

```
while    Condition do begin
         Statement
end;
```

whilst a var statement has the template

```
var
   Name : Type
```

Typing a reserved word, followed by space or tab, causes the template to be displayed on the screen. The placeholders which must be replaced by the programmer are underlined on a monochrome screen. Any attempt to type an illegal symbol in a placeholder is flagged as an error. Undeclared variables may be typed within statements, but these will then be highlighted in reverse video ,and a message displayed on the top line of the screen. The program is held within main memory in a tree format. Each node of the tree consists of a Pascal construct, with corresponding descendent nodes. In the case of a while statement, the node has two subtrees, one for the condition and one for the statement. The condition node must be a single Boolean expression, which is represented as a tree, whereas the statement subtree can consist of a list of statements so is said to be a list node. Each subtree has its own subtrees, and so on. It is this tree structure that ensures that only a syntactically correct programs may be written.

The second component of ALICE is the interpreter and is used to execute the program held in tree format. It is at this stage that some semantic errors are detected, an example being incorrect type expression in a condition.

The third component is the debugger which assists the programmer in detecting logic errors. This is done by permitting the program to be executed one step at a time, the setting of 'breakpoints' to stop execution on a particular statement, and the cursor to zip through the code on the top half of the screen whilst the results appear on the lower half.In addition, there is an immediate mode which allows values of variables to be displayed at any point where the program has stopped.

The final component is the help facility which like all of ALICE is menu driven and very extensive. Not only is specific help given on aspects of ALICE and Pascal itself, but it also allows such queries as 'What can I type here?' and 'What was my last error?'

As mentioned ALICE is menu driven with the main menu displayed across the top of the screen, and the operations and the other menus are invoked by using function keys. F1 invokes the interpreter to execute the program continuously, and F2 to single-step it. Other function keys invoke different menus and a full list is given in the Appendix.

The file menu gives access to the MS-DOS file system including DOS commands. A request to load a file causes a list of ALICE format files to be displayed; these files have the DOS extension .AP. No other file types are displayed. Selection from a menu is made either by a single letter or using the cursor keys. For the more experienced programmer, ALICE allows some functions to be invoked with the ALT key in conjunction with other keys.

Program development

When users first start to code a Pascal program using the ALICE environment, they are presented with the template for a program.

This consists of

```
program Program-Name (input,
output);
{Comment that says what the routine
does}
Declarations
begin
Statement
end.
```

Cursor keys can be used to move the cursor anywhere on the template, and the Control Key combined with left or right arrow keys moves the cursor forward or backwards to the nearest placeholder. Provided that the algorithm is well specified and proper preparation has been carried out before starting to code, using ALICE is very straightforward. The cursor is positioned over the 'Program-Name' placeholder and the required name is typed; this will replace the placeholder. Pressing the RETURN key moves the cursor to the comment line. Again typing a comment replaces that placeholder. This process is continued by typing in Pascal reserved words to give the required template, and then merely replacing the placeholder. Should typing mistakes be made,and not discovered until after the placeholder has been replaced, the.. the Undo command (CONTROL-U) which goes back to the previous step, may be used; there is also a Redo command (CONTROL-R) which is the reverse of Undo.

If procedures are being used, then typing the reserved word procedure gives the template procedure

```
Proc-name (Parameter);
{Comment that says what the routine
does}
Declaration
begin
Statement
end;
```

This can be completed in the usual way. Invoking the procedure by typing its name causes the parameter list to appear as placeholders. For example, the procedure with declaration

```
procedure SUB1(var x : real ; n :
integer);
```

when invoked by typing SUB1 would produce the template

```
Sub1 (var x : real, n : integer);
```

This removes from the programmer the need to remember the variable types and whether they are called by value or reference.

One further aspect is the ability of ALICE to determine where certain types of statement should go. Often programmers, particularly those who are just learning, forget to declare a variable. With ALICE, any attempt to use an undeclared variable produces a syntax error message but allows the variable to stay. At that point, typing var causes the corresponding template to be displayed in the appropriate declaration section and the necessary variable can be declared.

On completion of coding, the interpreter is invoked by pressing Function Key F1; the program runs until completion or until an error is detected. A single keystroke returns the user to the ALICE editor.

Once a working program has been produced it may be saved as a text file, rather than in ALICE tree format, and compiled using a compiler system such as Turbo Pascal, for greater efficiency. With ALICE, the program development may be described as in Figure 2.

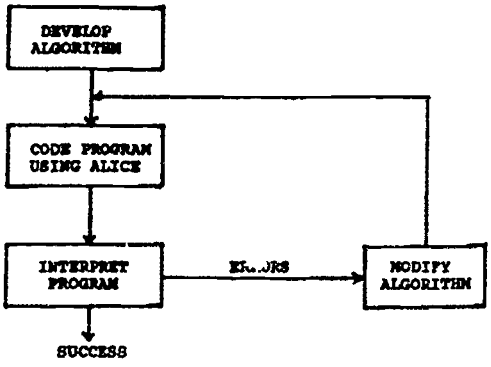


Figure 2

Difficulties encountered using ALICE

Given the nature of the ALICE system and its reliance on templates, it is quite difficult to make changes to the program structure without dismantling the code to some extent. All statements within control structures are nested within begin and end even where there is a single statement following then else or do. This means the user has to be aware of which template generated the current begin—end block. Figure 3 shows section of Pascal code produced using ALICE templates; (a) is correct and for every value of NEXTSCORE, the procedure PROCESSCORE is executed, whereas in (b) PROCESSCORE is invoked only for the final value of NEXTSCORE on each line.

```
while not eof do begin
     read (NEXTSCORE);
     if eoln then begin
          readln;
          end;
     PROCESSCORE
end;
```

(a)

```
while not eof do begin
     read (NEXTSCORE);
     if eoln then begin
          readln;
          PROCESSCORE;
     end;
end;
```

(b)

Figure 3

It is very easy to produce (b) when (a) was intended, because on completion of the block following them, the user must move the cursor out of the block and press return in order to generate the next placeholder. First-time users often overlook this.

The code resulting from ALICE can be difficult to read because of the large number of begins and ends. The practice of appending a comment to an 'end' is not available conveniently in ALICE, as comments, apart from those attached in declarations, must appear on a line by themselves.

Changing existing programs can cause some problems all of which are related to the tree format in which the program is held; two examples are given:

1. to move sections of code, it is necessary to:

- select the section to be moved, and move it to workspace (an area of memory;
- select the section again and delete it
- move the cursor to the position to which the code is to be moved, and retrieve it from the workspace.

2. to edit an arithmetic expression or an identifier, it is necessary to select it and invoke the edit function. This could involve three key strokes.

For obvious reasons, one function that ALICE cannot perform is to create data files directly at the keyboard. This means that if a file processing program is to be written, then it is necessary to write programs which will create the data files to be processed. From a teaching point of view this could be seen as an advantage, but with first time users it can cause confusion, particularly as programs must also be written to display these files.

A minor problem is that a program in tree format when saved in a file cannot be displayed except by the ALICE system. If there is a machine malfunction it causes the program to be stored incorrectly, then the whole program is lost. ALICE does not provide automatic backup.

Programs may be stored as text files and subsequently recovered from within ALICE using the text to tree format conversion program called APIN. This program is invoked automatically for a program file with extension .PAS, but unfortunately the user must remember the file name as it will not be displayed by any ALICE menu.

Conclusions.

The most noticeable difference between developing programs in ALICE and any other environment is the amount of typing required by the user. With ALICE this is reduced considerably because of the use of templates, and this usually leads to faster coding. It has extensive help files, is menu driven and protects the user with such prompts as 'Your file has not been saved', giving the option to return to ALICE to save it. Being language intelligent, syntax errors and type compatibility errors are detected as they occur allowing immediate correction. This lulls the first users and some experienced ones into a sense of complacency, to the extent where a number do not bother to learn Pascal syntax rules. It is interesting to note that most Pascal programmers who have learnt in a more traditional environment do not like ALICE at first. They complain it is restrictive and difficult to use, although the majority eventually accept it has much to offer.

Its major advantage is that it allows problem solving techniques to be learnt independently of both the operating system and the coding process. The latter process is relegated to its rightful position which is merely as a means of getting the algorithm into the computer.

From the program development point of view, the most important process is the design of the algorithm. In the course taught in this School, Nassi—Shneiderman diagrams (Nassi & Shneiderman,1973) are used, mainly to impose discipline on algorithm design. From these diagrams, the required procedures and variables, both local and global, may be easily identified and from that point coding using ALICE is relatively straightforward.

References

Looking Glass Software (1986). *ALICE : The Personal Pascal User Guide.* Toronto:

Software Channels Inc.

Nassi, I. & Shneiderman, B.(1973). Flow-chart techniques for structured programming. *SIGPLAN Notices, 8*(8), 12-26.

Appendix

Key	Function
F1	Run the program
F2	Single step
F3	Delete menu
F4	Insert menu
F5	Change menu
F6	Debug menu
F7	Menu of menus
F8	File system and DOS menu
F9	Help menu
F10	Select item
ShF9	Miscellaneous

Michael Newby is a lecturer in the School of Computing and Quantitative Studies at Curtin University. He holds a BSc(Hons) in Mathematics from the University of London and an MSc from the University of Bradford. His current research interests are into the development of software engineering tools and the use of such tools in teaching programming.

The powerful ideas of educational knowledge engineering

Neville Stern
Senior Lecturer, Information Systems Group Faculty of Business
Royal Melbourne Institute of Technology

Building a knowledge base in practice instructs the student, in more than the domain-specific knowledge which may form its contents. Conceptual modelling is instructionally active in the culturally significant area of the *construction of knowledge*, at a level at once distinct from and inclusive of the *transmission of contained knowledge*. The powerful ideas of educational knowledge-engineering thus have to do with *practical epistemology*.

Powerful ideas

With the usual genuflection in Papert's direction (1980), powerful ideas are educative ideas: they emerge from simple, concrete experience to become generalisable and applicable in contexts far removed from their original circumstances (Yazdani, 1984a). They are characterised by a quality of personal enablement. The power of the ideas, in cognitive terms, lies in the way they form the basis for *assimilation* of knowledge. Hence, in a well-known passage from *Mindstorms*:

> Slowly I began to formulate what I still consider the fundamental fact of learning: Anything is easy if you can assimilate it to your (mental) collection of models. If you can't, anything can be painfully difficult....The understanding of learning must be genetic. It must refer to the genesis of knowledge. What an individual can learn, and how he learns it, depends on what models he has available. This raises, recursively, the question of how he learned these models. Thus the "laws of learning" must be about how intellectual structures grow out of one another, and about how, in the process, they acquire both logical and emotional form. (Papert, 1980, p. vii)

At the risk of pointing out what has been looked at many times, it is notable that Papert doesn't simply say "learning is genetic". It is the *understanding* of learning that is genetic. The power of powerful ideas has a metacognitive foundation: they have something to do with the consciousness of a construction to acquired knowledge, and with the corresponding consciousness of there being a process to that construction.

The powerful idea

All of this is by way of introduction to what strikes me as the "powerful idea" in knowledge-engineering, a commercial and product orientated technology of elicitation, representation and machine-acquisition of knowledge. The idea is simple: one learns by developing and modelling one's conceptual structures in a knowledge-based system which performs the role of an interactive, reconstructable medium. In effect, the medium plays the student.

Using the technology of knowledge-engineering to build knowledge-based systems is a process which can educate the builders.

Feigenbaum (1984) has commented that

> In the end it may be irrelevant that a computer program is available to be an "intelligent assistant" The gain to human knowledge by making explicit the heuristics of a discipline will perhaps be the most important contribution of the knowledge-based systems approach. (p. 50)

Knowledge-engineering has developed as a commercial application of artificial intelligence (AI). The interesting point is that it has frequently resulted in a learning benefit for those involved. An extension and modification of human expert knowledge, and more efficient industrial practice are two examples of such a non-computer-based and commercially significant result. In some cases, the knowledge-base has been discarded in the light of what may have been thought of as a side-effect, as Feigenbaum (above) has suggested.

Knowledge engineering can have a generalised educational application, therefore, which can be distinguished from conventional methods of computer-assisted instruction. Some researchers have touched on aspects of the application, mainly in connection with young children. For example, Bigum (1986) has commented:

> ...the building of even very simple knowledge-based systems by children i.e. by having them make their knowledge and understandings explicit, and conform to the requirements of a shell, may provide their teachers as well as the children themselves with an opportunity to expand and more importantly to critically question their knowledge.
>
> Building a knowledge-based system will not be important because of the finished product. The educational worth of the enterprise will be entirely located in the processes children will probably go through in order to construct such a system. The "finished" system may have a value in provoking other learners to reflect about their own understandings, but that is an incidental plus. (p.180)

And in the same context of primary education, Cumming and Abbott (1986) have made a case for using implementations of logic programming:

> ...children ...can look inside (expert systems in micro-Prolog), make additions and modifications and eventually, write their own. Teachers also should find it easy to use and modify programs and write their own. So the range of possible educational activities is extremely broad. The suitability of micro-Prolog as a language for use by children and teachers, and as well for the implementation of educational (expert systems) may be its single greatest strength from the point of view of education. (p. 133)

Few apart from the author and some Australian colleagues (Stern & Adams, 1985; Stern et al, 1986; Stern, 1986a, 1986b; Gilding, Stern & Forrester, 1986) have referred to the learning effects of the process as a generalised and comprehensive approach to teaching and learning, particularly at the tertiary level. Tertiary vocational and professional education tends to be dominated by what can be described as skill and knowledge *transfer* styles of instruction, which seems to many to be suited to mechanical tutoring methods. A knowledge-engineering approach, however, can be easily integrated with standard and available educational materials and instructional methods, including continuous and examination-based assessment. It also represents an interesting point of departure.

An illustration by projection

A simple illustration of what such a departure may lead to can be provided in the form of a projection. In conventional tertiary educational methods, for example, students write essays and sit examinations to demonstrate their knowledge. In a knowledge-engineering approach to teaching and learning, students will also build knowledge-bases with available materials. The knowledge-bases will be examined and tested. The immediate purpose will be to have the student-built system demonstrate a certain requisite knowledge content.

The process itself will have a broader, more important purpose. The resultant knowledge-bases will be convincing models of human knowledge. They will be able to be taken as material evidence of a human depth of understanding of the knowledge, originating in their creators. That understanding will have been gained through the process of analysis and construction, and through critical examination of the adequacy of both the knowledge and its representation in the model. The value of the approach will thus inhere in process as well as product, and in personal learning as distinct from rote memorisation and recall.

Learning as construction

The process of building knowledge-bases could therefore make a good teaching and learning method by placing students in the role of knowledge engineers. Connections can be made with important bodies of learning theory and application, in the broad context of using computers for education.

Learning can be viewed in cognitive terms as fundamentally a construction process, whose results and whose building blocks are conceptual structures. Intelligent systems which incorporate machine-learning have relied on this view, to model student behaviour during computer-assisted instruction. In terms of constructive human-directed learning processes, however, *learning by building* these structures and models in an external interactive and editable medium is an extrapolation of that constructivist view. It is a synergistic activity of reflection, both internally and externally.

Instruction could be described, particularly in the context of mechanical tutoring systems, as the methods of building conceptual structures in students. It includes strategies to bring the student's mental model of knowledge into some correspondence with the machine's. The terminology of engineering, and of knowledge-engineering, is frequently associated with the instruction of people in this way. The goal of the activity is the communication, transfer and registration of well-defined quantities of representable structures of knowledge. The reception of knowledge by the student is acknowledged and verified by simple recall, and other responses to questioning requiring elementary explanation and demonstration facilities. The parallels with data-communication are obvious,as are the connections with the activity of knowledge-engineering.

Reversing the direction of the effort of registration and correspondence, students who build models of their own knowledge in intelligent systems are creative instructors of both the machine and themselves. This activity of instructive modelling can provide students with the role, insights and learning experience that normally accrue to their teachers and other authors of course materials..

Broadening the focus of attention in knowledge-engineering

The focus of attention in applying these ideas educationally has to be broad enough to take process as well as product into account. Bigum's comments earlier emphasize the process aspect, effectively freeing up the dependence we may feel, or want to feel, on sophisticated technology. The technology is with us, however, and there is no reason why the machine-based product of educational labours should be underrated. At the tertiary level in particular this prospect is encouraging.

But knowledge-engineering is a relatively new, commercial and product-orientated activity whose constituent processes and technology are chagrining, and are not yet well understood. The sole focus of commercial attention and incentive to date has been the development of a finished product. Both the processes and the product tend to have been regarded as proprietary. The focus, purpose and the concealment from public scrutiny of the activity tend to deflect attention from the human learning effects, and to place educational efforts to use the technology in the toy category.

A recent comment (Jones, 1987) from a professional knowledge-engineer in a journal under the heading "Commercial AI: Expert Systems for Blood: Building a Business" nicely sums up the issues of learning benefits and proprietary silence:

> A problem for commercial clients in particular is that the consultant (knowledge engineer) *gets to know a great deal about the customer's business.* This drives some organisations to attempt to grow their own in-house groups....The secrecy with which some organisations wish to surround their expert systems activities leads to...(a) frustration. It is often impossible to get permission to talk about the customer and/or the application to others. When embarking on an area like expert systems applications, it is pretty obvious that having good reference sites gives added credibility both to the contractor and to the technology itself.
>
> It is our view that endlessly talking about the great systems of the past (MYCIN etc.) gives a misleading view of what is currently being achieved in the field. However, the number of case histories that are available for general viewing is sadly limited. (p. 10) (my emphasis)

A closer examination of the process, in educational contexts, could lead towards a more general application of the technology, as well as firmer underpinnings for its use in specific areas.

The strengths (and weaknesses) of Intelligent Tutoring Systems

The powerful idea of educational knowledge-engineering is an epistemological idea, having to do with the construction of knowledge. How does the process compare with tutoring systems, and their central idea? Yazdani & Lawler (1986) summarise the application of artificial intelligence in Intelligent Tutor Systems:

> ITS have a central objective of communicating some knowledge through computer facilities which may employ domain-specific expertise, error analysis, and user modelling. Their user-oriented intelligence is controlled by instructional strategies which present (the user) with (domain-specific) problems and then test for understanding of the knowledge, generally conceived of as a collection of methods for problen solving, as well as domain-specific knowledge.

They state that the strengths of ITS derive from "good definition and... completeness" in the sense that an ITS

> will have a well articulated curriculum embodied in its domain expertise and an explicit theory of instruction represented by its tutoring strategies. This completeness permits an ITS to package existing expertise and focus on the

novelty, which is the use of mechanically embodied sets of rules as a tool for instruction. Because an ITS can be well defined for a given curriculum, the achievement of its goal, in principle, can be unambiguously evaluated.

The weaknesses of ITS "set against these considerable strengths" are, they state:

> ...inadequate complexity in what the user knows, how the user learns new knowledge and what could be considered an appropriate instructional theory for complex minds.

In terms of the knowledge-engineering approach I am advocating, such weaknesses are not necessarily redressed by increasing the internal complexity of ITS. The argument is that an appropriate instructional theory for complex minds needs must take a knowledge-engineering approach into account, on the grounds of the self-constructive character of mind..

Making ITS more complex internally

The attraction and challenge of an ITS is to get it to behave as a flexible instructor, sensitive to the "complexity of the instructional situation" (Yazdani & Lawler, 1986). The focus is therefore on the issue of designing machine-learning and other expert systems internal to ITS to support this intended role. The idea that an ITS can be improved by incorporating several complex knowledge-bases, and "sites of intelligence" (Richards 1985, 1986), which are separate and distinct from the learner, and set in a particular relationship to the learner's activity, has attracted considerable research interest. One feature of this approach, when human learning with an ITS is discussed, is that it is predicated upon the richness and complexity of machine-learning in the student model internal to the ITS.

Yazdani & Lawler (1986) take an example from Brown and Burton's work (1978) in teaching arithmetic, which requires the ITS to analyse and respond to "bugs" when a "child's logic represents a deviation from the standard addition algorithm". They go on to describe the resulting issue as follows:

> There will surely occur cases where the student uses representation beyond the current scope of the system. What should the system do when it determines that it does not, and even more cannot, understand what the student is thinking? The systen should have available an extensib.. repertoire of representations. No system which is too rigid to learn should be called intelligent.

They propose that the intelligence of the ITS, therefore, should consist in large part of being able to learn by experience, rather than by being told - a crucial step which effectively begins (but of course does not necessarily seal) the process of removal of such intelligence from the direct control of students:

> While people learn more from experience than from being told, for computers the opposite is more nearly true. Computers can learn from experts today; for use in education, an ideal instructional system should be able to learn new representations applying to its domain of primary expertise even from non-experts, as when the students are introducing some alternative way of thinking - however non-standard that way may be.

It is central to the knowledge-engineering paradigm that what Yazdani & Lawler say should, ideally, be true: that "computers can learn from experts today". But it is *how* they learn which is vital. To develop an instructional theory for the knowledge-engineering paradigm one needs to give students the latitude to *instruct the system*, and to exploit the system's learning capabilities in the same way that ordinarily a teacher might a student's. If this latitude is in some way reduced, by assuming for example that it does not exist, then of course the basis for the paradigm is weakened.

Concern about the knowledge-engineering approach: it's too easy

Since one of the components of the knowledge engineering approach is an experiential approach to learning, the real difficulty of accounting for the *what* and the *how* in the experiential process should not be underestimated (as Yazdani & Lawler, 1986 in fact warn). Following this line of argument, it can be said of many tutoring systems that it is time to be suspicious when they assert a narrowly specific mechanism of *what* and *how*.

Few would regard drill-and-practice or an overlay approach to student modelling as adequate expression of how a complex mind learns. But there is in Yazdani & Lawler's arguments a demonstrable bias against leaving issues of instructional complexity of a tutoring system to the human using it. What a human does to the system can be, in a curious twist to the argument, derogated because a). it is too easy and b). one doesn't understand it because it is too complex. As an example of this bias, here is Yazdani and Lawler's criticism of learning by building a knowledge base:

> Knowledge based systems, with rules stored separately from the processes which use them, have proven their usefulness because their *very lack of organisation* permits the addition of new rules without reprogramming; if, on the other hand, the organisation and reorganisation of knowledge in the mind is the central need in effective education, thinking of learning as 'adding another rule to the database' may be counter productive (p. 74) (italics my emphasis)

If adding another rule to the database is inadequate as a view of learning, and as an action has no educational significance, then the knowledge-engineering paradigm is on shaky ground indeed. But mechanical reductions of the learning process however should be recognised for what they are. It is not simply the mechanical addition of another rule that constitutes learning within a educational knowledge-engineering paradigm. It is the consistency of that new rule in terms of the organisation and knowledge-representation discipline of the knowledge-based system that really matters.

It is easy to seize upon the *editability* of a knowledge-base, and to object to it in isolation from the formal organisation, and therefore the testable behaviour of the whole. Logic, for example, has been from the time of Aristotle a discipline of building consistent and valid discourses. Adding another rule to a logic program is equivalent to adding another statement to a logical proof. The consequences, and the significance of such an act, are in either case the same.

The redress for the problem of a rule added at random lies in testing the system itself for adequacy and consistency. Notwithstanding the mystery of how a student learns through experience internally, we will have some measure of the success of that process if that student can bring the system to the required standard. To rephrase Yazdani and Lawler: *thinking of learning in this way is productive if it can be seen in terms of the organisation and reorganisation of knowledge in the mind*. In this light, some of the "considerable strengths" of ITS can be incorporated in the knowledge-engineering approach: there are clear processes to follow, and well-defined externally assessable products as a result.

Concerns about the knowledge engineering approach: it doesn't instruct

Interest in ITS can also be fueled by concerns, often provoked by the experience of badly managed alternative approaches, about about guidance, control and instruction. Here is the viewpoint of a critic of Logo usage (Cumming, 1985):

> Children left to learn in an unstructured, do-it-yourself hands-on way sometimes generate for themselves quite bizarre mental models. In contemporary writing on education there is a strong theme that such laid-back, *laissez-faire* approaches are to be applauded. Partly this is an understandable reaction to the worst rigidities of teaching machines, partly it is a rationalisation in the face of the great difficulty of writing good CAL (Computer Assisted Learning) materials, partly it is a healthy affirmation of choice, relevance and motivation. But we must be vigilant! Encouraging children simply to go to it may be fashionable, but will for some children lead to serious misconception. We must not sidestep the responsibility of curriculum designer and teacher to choose and structure and guide: helping a learner build a good mental model is too important for this.(p.9)

The concern is aggravated by empirical results such as these (Gentner & Stevens, 1983):

> ...all too often there is no correspondence among the conceptual model of the system that guided the designer, the system image that is presented to the user, the material in the instructional manual that is taught to the user, and the mental models of the user. (p.13)

The resulting dilemma several authors have described as follows:

> One wants to support a child but not the development of a wrong theory...we want to support the child's commitment to his authentic point of view and its creative application; at the same time we want to modify that view.
> (Yazdani & Lawler, 1986)

> ...the central problem of humane education is how to instruct while respecting the self-constructing character of mind. (Lawler & Papert, 1986, p. 69)

In the knowledge-engineering approach, the differential between human cognitive models of knowledge, and those representable in a machine can however be used as a dynamic of human learning. It does not exclude issues of human complexity, nor discount the useful aspects of building a model consistent enough to run in the machine. The cause for concern arises when one tries to bring people into line with machines, not the other way round. Papert (1980) made this point in a broader social context:

> Consider another example of how our images of knowledge can subvert our sense of ourselves as intellectual agents. Educators sometimes hold up an ideal of knowledge as having the kind of coherence defined by formal logic. But these ideals bear little resemblance to the way in which most people experience themselves. The subjective experience of knowledge is more similar to the chaos and controversy of competing agents than to the certitude and orderliness of p's implying q's. The discrepancy between our experience of ourselves and our idealizations of knowledge has an effect: It

intimidates us, it lessens the sense of our own competence, and it leads us into counter productive strategies for learning and thinking. (p. 192)

It is hard to see the knowledge contained within an ITS as anything other than having the "kind of coherence defined by formal logic", irrespective of its degree of sophistication or complexity. Similarly, irrespective of the complexity or sophistication of the strategies of instruction, the central irreducible objective of tutoring systems remains: to transfer a quantity of knowledge. The problem is to avoid such a "counter productive strategy for learning and thinking" and yet still to guide; to instruct so that the learner constructs, rather than forcing the issue to the point of mechanical communication so that the learner remembers.

My argument is that the best resolution of that problem is to be found in a serious-minded application of knowledge-engineering in education. We have nonethless to meet an important objection to computer-based learning environments in general, that a "model-based epistemology of instruction does not exist" (Yazdani & Lawler, 1986). It is argued in this view that in a modelling system there is no set of instructional techniques which derive their justification and purpose from a specific and empirically verifiable view of the construction of knowledge.

The question is begged if a narrow view is taken of instruction, the direction of communication, and of the separate roles of teacher and learner. If teaching can be connected with learning, then there are many strategies available to the student who does the instructing of the machine. The knowledge-representation scheme of the system, and the means of programming in it then connect in the answer to the simple question of: in what way does the student tell the system what to know? This is related to the frequently asked question of: how does the system learn? The simple answer is of course, through what the student teaches it, and not necessarily through what it can infer from the student's behaviour.

In terms of who or what does the shaping, one should also consider the scope of the environment within which the student can construct things and get them to "run" satisfactorily. Here the image of knowledge is as a *shape* to an explorable *space* of "things to do". This space is a microworld: a bounded environment populated with computational objects, having some representational mapping to activities in the "real world" (Adams, 1986).

The merits of this view include the recognition of a hierarchical and recursive composition to microworlds. The designer of such a world can include in it the objects and tools to make and to contain other worlds. Hence the role of designer can be transferred to the designed, or more precisely to the user within a designated space. Such a microworld, in other words, can be viewed and presented as just another computational object. The concept copes with (fragmentary) collections of microworlds, and of computational objects within worlds.

The major weakness of a simple discovery-orientated learning strategy, as we have noted at several points in this thesis, is its openness to aimless wanderings. This objection can be partially met by acknowledging that explorations of microworld spaces can be directed by teachers, and by job or life-related requirements to do something specific. An example of the latter could be the building of industrially useful knowledge-based systems using tools within a knowledge-engineering environment.

Nonetheless the apparent lack of an innate instructional strategy, belonging to the microworld and based on some view of epistemology, is not fully dealt with by reference to external direction. The argument is further weakened, in the knowledge-engineering example, by the very generality and scope of the working environment. What does one learn from a blank sheet of white paper, or a word-processor for instance?

One must therefore look for instructional strategies in what the modelling environment does to the student as an active consequence of its design and character, not simply in the constraints, more or less, placed upon him or her by external agencies or the scope of the modelling environment. To do justice to the notion of instruction, it must come *to* the student. What he or she does, whether directed or not, should give back something. The reflection in the mirror has to teach, in other words.

Instruction through practical epistemology

A model-based system teaches something very concrete and specific about knowledge. Its real "instructional goal" is to communicate a number of personally and socially significant epistemological statements. These are that human knowledge is at once

- representable;
- constructed;
- limited and
- editable.

It is easiest to see these communications as instructive when the student is given a knowledge-based system, for example, which already has some significant and relevant knowledge modeled within it. But there is also a strong basis for the argument in a concept-modelling shell itself, a message let us say, in its medium. In a sense, the ability to *run* a concept model is an instructive and live demonstration of the answer to the question one frequently asks of other people: how come they can think (or say) that? This is *practical epistemology.*

Each epistemological statement has important repercussions in terms of our understanding of the cultural and personal bases for knowledge, and our own views about how it is possible to alter knowledge, and hence learn. The active instructional purpose of a model-based system therefore is simply to make it possible to learn those important things about knowledge by building and "running" models. The point is at its sharpest in an interactive concept-modelling system. The instructional strategy of such a system therefore cannot be seen in the same terms as that of a linear or branching or iterative drill. Its curriculum is the construction of knowledge itself.

Conclusion

In summary, the powerful ideas of conceptual modelling have to do with practical epistemology. Building a knowledge base does in practice instruct the student in more than the domain-specific knowledge which may form its contents. Conceptual modelling is instructionally active in the culturally significant area of the *construction of knowledge,* at a level at once distinct from and inclusive of the *transmission of contained knowledge.*

I have argued that, within the knowledge-engineering approach, a putative gap between a student's mental model and the model of knowledge internal to an ITS can be exploited as a dynamic of human learning, rather than seen as a problem for the machine to solve. Instead of bringing the student's knowledge into a correspondence with system knowledge, and testing for that direction of correspondence, the reverse becomes true. Both the machine-based and the personal models will undergo dynamic alteration as the one, and by implication the other, is brought to a new designated and testable standard of knowledge.

The role of a (human) teacher, and the responsibilities to guide and instruct are clearly not undervalued or affected by the alternative knowledge-engineering approach. The communication of an existing and contained body of knowledge, with which the student is to work and be equipped, is not necessarily impeded by doing this. It is not impossible that students' knowledge will remain unmodified by this process of exploration, construction and instruction of the ITS. Normal evaluation and assessment procedures in a tertiary level institution are not disregarded. A constructive form of independent learning becomes possible, and particularly valuable in periods of financial constraint. Support for the student's activity can be provided in a variety of forms: available knowledge-based materials, such as lectures, tutorials and textbooks; and content, procedural and process advice ˊ ɔm human teachers. It can then be argued that a student is in a better position to develop a deeper understanding of knowledge that may otherwise have been rote-learnt.

The approach leads me to reverse an earlier quotation from Yazdani and Lawler (1986):

> (ITS) user-oriented intelligence is controlled by *students'* instructional strategies which present *the tutor* with (domain-specific) problems and then test for understanding of the knowledge, generally conceived of as a collection of methods for problem solving, as well as domain-specific knowledge.[my emphasis and alteration]

The "considerable strengths" of ITS as described by Yazdani and Lawler also stand up well to this re-viewing and rephrasing. Once a student has finished building a particular system that passes all the tests an assessor can devise, it then "will have a well articulated curriculum embodied in its domain expertise". It will also have an "explicit theory of instruction represented by its tutoring strategies", except that here the tutoring strategies are those of construction, and the explicit theory of instruction is based on its connection with construction. And finally, as expected of an ITS, the "achievement of its goal, in principle, can be unambiguously evaluated".

There is an entertaining sense, in the foregoing, of doing things to tutoring systems that they were supposed to do to you. More seriously, though, connections are possible between the knowledge-engineering approach and the use of general computational environments which also span the four standard models of computer usage in education. Their major characteristics can be described as: an openness to user access, and a continuity between what the user does with them and the means of their own construction. An important example is afforded by Boxer (di Sessa, 1982).

Bibliography

Adams, T. (1986) *Towards a Theory of Microworlds,* in Salvas & Dowling (1986), 312-320.

Bigum, C. (1986) *Knowledge-Based Systems in Education: The Fourth Tool?* in Salvas & Dowling (1986), 176-182.

Cumming, G. (1985) Mental Models - the world inside our heads, *COM3* 11(1), February 1985, 7-9.

Cumming, G. & Abbott, E. (1986) *Exploiting Expert Systems in Education,* Proceedings of the First Australian Artificial Intelligence Congress, Melbourne, November 1986, 238.

di Sessa, A. (1982) *A Principled Design for an Integrated Computational Environment,* Laboratory for Computer Science, MIT, Technical Report MIT/LCS/TM-223, July 1982.

Feigenbaum, E. A. (1984) *Knowledge engineering: the applied side* in Hayes & Michie (1984), 37-55

Gentner, D. & Stevens, A. L. (1983) *Mental*

Models, Erlbaum, New Jersey, 1983. .

Gilding, A., Stern, N. R. & Forrester, C. (1986) *Students as Knowledge Engineers*, Proceedings of the First Australian Artificial Intelligence Congress, Melbourne, November 1986.

Hayes, J. E. & Michie D. (1984) (eds) *Intelligent Systems: the unprecedented opportunity*, Ellis Horwood, Chichester, 1984.

Jones, R. (1987) Commercial AI: Expert Systems for Blood: Building a Business, *Professional Computing* 22, Australian Computer Society, April 1987, 9-10.

Lawler, R. & Papert, S. (1986) *Intelligent Videodisc Applications for Pre-readers*, Journal of Educational Computing Research, 1(1), 1985, p. 67.

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas*, Harvester, 1980.

Salvas, A. D. & Dowling, C. (1986) *Computers in Education: On the Crest of a Wave*, Proceedings of the 1986 Australian Computer Education Conference, Computer Education Group of Victoria, Melbourne August 1986.

Stern, N. R. & Adams, T. (1985) *Teaching Prolog to Students of Business*, Proceedings of the First Pan-Pacific Computer Conference, Australian Computer Society, Melbourne, 1985, 1015-1030.

Stern, N. R., Hinrichs, J., Lukaitis, S., & Kimber, D. (1986) *New Directions in Business Technology Education*, Proceedings of the 17th Annual Computer Conference, Australian Colleges of Advanced Education, Darwin, July 1986, 72-78.

Stern, N. R. (1986a) *Prolog as a Mind Tool?*, Proceedings of the First Australian Artificial Intelligence Congress, Melbourne, November 1986.

Stern, N. R. (1986b) *Prolog and Education*, Proceedings of CALITE 86, Adelaide, December 1986.

Yazdani, M. (1984) (ed) *New Horizons in Educational Computing*, Ellis Horwood, Chichester, 1984.

Yazdani, M. (1984a) *Artificial Intelligence, Powerful Ideas and Education*, Working Paper W138, Department of Computer Science, Exeter.

Yazdani, M. & Lawler, R. (1986) Artificial Intelligence and Education: an overview, *PEGBOARD*, 1(1), Exeter, Spring 1986, 68-74.

Bandwagons, hearses and other vehicles: The useful fate of Prolog in education

Neville Stern
Information Systems Group, Faculty of Business
Royal Melbourne Institute of Technology

The useful fate of Prolog in education is to open and join the way to educational knowledge-engineering, within a general framework of conceptual modelling. Together with work in wide and sometimes apparently disconnected areas, it contributes to the possibility of a new approach to teaching and learning that incorporates many of the strengths and ameliorates some of the weaknesses of its precursors.

The confluence of a number of strands

Building knowledge bases for educational purposes is a useful approach to teaching and learning with the technology of Artificial Intelligence. This paper briefly reviews a number of educational and knowledge-representation research projects which have contributed to, or can now be linked under the heading of, educational knowledge-engineering.

Work with **Logo** in the development of computer-based modelling environments, and with **Prolog** in a movement towards the educational construction of knowledge-based systems, are two important contributory strands. **Concept mapping** and **conceptual graphs** are examples of two distinct activities, the one educational and the other technical and theoretical, which can also be connected in this way.

Prolog and the idea of educational knowledge-engineering

As one of the important strands converging on the idea of educational knowledge-engineering Prolog (Programming in Logic) has induced a shift of interest in this direction by both its strengths and its weaknesses.

Amongst its strengths in principle:

- it is an easily available knowledge-representation and knowledge-base building tool, with elementary logic as a discipline of construction within a broader rule-based methodology;
- it has encouraged the sense of an 'organisation to knowledge' which children can easily appropriate and master;
- it has also facilitated, to some extent, the placing of novice students in the
- role of builder of the expert system;
- its major strength is its support in principle for the representation of concepts and a consequent conceptual modelling activity.

Amongst its weaknesses in practice:

- an otherwise appropriate view of logic as a discipline "in its own right" leading to the promotion of logic as a single important "backbone" (Ennals, 1984) to more general areas of knowledge;
- hence, conflict with the ways in which people actually 'know', and other more flexible means of knowledge representation;
- an overemphasis of the value of declarative as opposed to procedural knowledge, which implies a static or finished state to knowledge;

- a consequent assumption that, given a complete specification of a problem, problem-solving is to be carried out by the machine, overlooking whence the specification is to come;
- inherent difficulties in declarative logic of dealing with change of state and dynamic alteration of the knowledge-base;
- in the controversy over procedural and declarative interpretations of a logic program, a confusion of the procedural interpretation with process issues to do with actually developing a program in logic;
- hence an under-valuing of the dynamic, modelling and prototyping processes of developing a knowledge-base in declarative logic;
- limitations and complexities in interface, syntax and environment militating against the construction by novices of substantial and non-trivial knowledge-bases;
- and, as a database and query language, it has tended to sidetrack researchers into using it as an unecessary substitute for other more general and perhaps more easily used database and query languages.

Detail of the arguments for these assertions can be found in Stern, 1986a, 1986b.

Its strengths and weaknesses may be summarised as: support for knowledge-base modelling, and the limitations of a purely logic-programming point of view. Both have contributed to a broadening movement beyond Prolog itself.

Indicators of transition

The Prolog education movement is in rapid transition. It could be said that it gained some impetus as a bandwagon from the faltering, in France in 1982, of Logo as an educational vehicle. Subsequent experience following promotion of Prolog and of logic programming as a language for children's learning has led to a revision of its role amongst a broad and richly varied set of educational vehicles to avoid a similar hearse-like effect.

As a sample of aspects of the change, it is instructive to review Ennals (1985) comments on a report evaluating the the Logo experiment at the Centre Mondial de Programmation. Ennals translates part of the report as follows:

> ...if one looks at the computer products of the pupils...one is struck by the uniformity of the results and by the wretchedness of the programs....(An) hypothesis would be that the computer facilities...are too limited and impose too much *computer* thinking (as opposed to simply thinking) on the programmers, and that the uniformity of the results (in Logo it is rare to see programs which escape from the *square* and the *house*) is only a direct reflection of the weakness of the means at their disposal and the inadequacy of such computational constructs (and these languages) for expressing the mechanisms of problem solving. (p. 59)

and goes on to comment:

> There had been a tragic mismatch between potential and expectation. In many cases...the specification documents describing the problems to be dealt with were clear and explicit (much of the text could be run directly as Prolog databases!) but the Logo Turtle was hideously inadequate to deal with the diversity of problems, particularly in the hands of unprepared researchers.

I have argued elsewhere the inadequacy of *argument* for Prolog as an expression of the "mechanisms of problem solving". There has been, if not a tragic mismatch of potential and expectation, a tension between what was and could be done with the language, and what was said about it. For example many promot~rs of problem-solving with Prolog (such as Conlon, 1985; Ennals,1984, 1985) were, in the midst of their assertions of the advantages of declarative interpretations of programs, as was said of Milton, of the Devil's party without knowing it.

Other researchers, such as Cumming (1986) have cautiously begun to raise issues of limitations, beyond those of syntax or interface, to Prolog as a general educational medium. At the same time some inherent strengths to the educational use of the language are acknowledged:

> The restrictiveness of Prolog's structures for knowledge representation limits its educational applications, but still allows a usefully wide range for exploration. The severity and precise nature of the limitation has yet to be determined. Most importantly, Prolog and EMITSI provide an environment that is very friendly for learners and can support a wide variety of activities, including using, modifying and building small expert systems. (p. 60)

Indicators of this transition towards a more general authoring and knowledge-base construction approach can most clearly be seen, for example, in the work and publications of PEG (the Prolog Education Group at Exeter). Workers in the group produced a number of authoring programs written in Prolog fo~ use at first in the teaching of history through modelling and simulation. The software included simple decision-tree modelling of historical choice and action, an adventure-game building system, and database construction and query systems (Briggs, Dean & Nichol, 1985) A major weakness in this work, as far as the promotion of Prolog was concerned, has been the extent to which other software, such as simple database and query languages, could have done what their Prolog-based software did (Stern, 1986a).

A recent PEGBOARD editorial (Autumn 1986) began by asking the question:

> When does a bandwagon become a hearse?...There is a danger of over enthusiasm, and making claims which cannot be substantiated, or which other factors can explain.

and concluded with:

> The key concept is that of learning *with* computers, where the computer is seen as a powerful tool for learning. Prolog has a triple part to play here; the teaching of logic programming in order to develop logical thinking, the writing of logic ...programs to develop understanding of a problem or a topic, and the use of authoring programs or shells to achieve the same goal. The writing of Prolog programs and Prolog shells are based upon analysis of the logical structures of knowledge within subject domains. They force the writer of a program, be it a simulation, adventure game or database to structure the knowledge which the program represents. Prolog can have universal applicability in the development of logically based thinking within the triangular structure of subject knowledge domains' conceptual, procedural and propositional knowledge. (Nichol, 1986, p.5)

The concluding sentence indicates a recuperation not only of Prolog's original educational aims (the "teaching of logical thinking") but of more general educational knowledge-engineering and knowledge representation issues (the disciplining of the "writer of a program...to structure the knowledge which the program represents"), in a context in which "propositional knowledge" is only one component.

In effect, it is no longer Prolog that has "universal applicability", despite Nichol's re-assertion in the context of the "development of logical thinking" above, but a more general educational approach using broader knowledge-representation and knowledge-base building tools. That construction environment, which of course must include a rule-based and declarative logic methodology, need not be restricted to it, as a review of commercial knowledge-engineering systems and architectures will confirm (for example: Friedland, 1985; Fikes & Kehler, 1985; Hayes-Roth, 1985; Genesereth & Ginsberg, 1985).

Examples of this kind of integrated and tool-kit providing environment abound in industry, and to certain extent in education for limited purposes such as the study of artificial intelligence or cognitive mechanisms (for example POPLOG: Sloman et al., 1983; Gray, 1984). Most examples do not focus on the educational effect of building knowledge bases themselves. Boxer (di Sessa, 1982) is worthy of mention in this context, because as Nichol reports (1986) it is beginning to attract some interest in Europe as an environment for learning. As an "authoring environment" it is described by PEG workers as a kind of next step, something which "combines the learning features of Logo, Prolog, LISP and Small-Talk in one environment" (Nichol, 1985, p.6).

Logo Modelling and Concept Modelling

Logo has done more than provide a springboard for Prolog. There is an important connection between modelling a set of concepts in a language like Prolog, and the body of modelling theory that has grown up around Logo.

While modelling in Logo tends to be associated with the graphic presentation of turtle-behaviour, there is a more general point. The behaviour of such models is controlled at the level of program statement, a textual and linguistic level that opens up more general possibilities of using a list-structured language for symbolic representation. It is at this level, that the most important link with Prolog exists, since developing a knowledge-base model in Prolog is essentially a process of testing and extending the meaning of statements in a concept-description language.

The assertions that a Prolog expert-system is a learning microworld and an object-to-think-with, and that Prolog is a computer-based learning environment for creating such microworlds, can be confirmed by reference to current work in microworld theory (Adams, 1986) and by a reading of Papert's *Mindstorms* (1980) from the particular point of view of modelling with concepts.

A learning microworld (Adams, 1986) is both an environment for the construction of models, and a working model itself - a recursive, hierarchical composition. Every constructed model possible within a given environment is defined by the (essentially linguistic) set of instructions which constitute its program. Expert-systems no matter how small or prototypical can be viewed as *microworlds-as-models*, with defined behaviour and scope of activity, created within an *microworld-as-environment*.

Concept models in Prolog display key features of the modelling process. One is able to examine the model's behaviour, circumscribed by the program that defines it, and to change the scope of its possibilities of behaviour by concrete means - that is, simply editing the program and re-running it. The 'closed world' of any Prolog program, within an environment that allows this world to be changed, matches the concept of a learning microworld very closely. The important point is that one can always edit that closed world within the environment supplied by the Prolog programming system.

An insistence on a declarative world in which nothing ever "happens" makes a problem out of this creative editing process where none really exists. Furthermore, simply adding procedural capabilities to Prolog as a programming language, or turning to procedural interpretations of a logic program, does not represent a full response to the issue of building conceptual models in and through time. For example, extending Prolog to include Turtle Graphics (Kowalski, 1984) does not extend the notion of a learning microworld in terms of the significance of the editing and modelling process.

The body of microworld theory that has grown up around Logo expresses the idea of the computer as ultimately a tool for thinking about thinking - an extension of the more particular instances of the computer as means of thinking about something. Building knowledge-bases expresses this idea perhaps even more neatly than building abstract mathematical structures in turtle geometry. A small Prolog prototype expert-system is explicitly a microworld for thinking about thinking.

Concept Mapping and Conceptual Graphs

The confluence of the tributary strands discussed above, with the experience in industry of the human-learning effect of engineering a knowledge-based product, can form the basis for a generalised and inclusive notion of conceptual modelling as both a learning activity, and a knowledge-base construction activity. There are paths to this idea, from two distinct and yet-to-be connected areas of research in knowledge-representation.

The first, *concept mapping* (Novak & Gowin, 1984) is an educational technique which has so far been done without computers. It derives from studies in cognition (Ausubel, Novak & Hanesian, 1980), and connects with knowledge representation techniques for computers, in particular semantic networks and more generally a formulation of concepts and their linkages in terms of connected graphs. Kay (1986) has described the use of computer-based concept mapping in an educational context, but restricted her interests to a means of representing knowledge in a form by which an intelligent tutor system could confirm that a certain body of knowledge had been successfully communicated to a student.

The second is in fact a significant generalisation of the first, but one in which all educational implications have so far been ignored. Computable conceptual graphs, and hence the idea of a conceptual graph processor which could include all the features of logic programming, are the work initially of Sowa (1984). This work is almost exactly contemporary with that of Novak and other researchers in educational concept mapping. Neither Sowa nor Novak acknowledge the existence of each other in their work, nor so far have their followers.

There is an obvious and missing connection to be made. The link is of course the educational knowledge-engineering approache, provisioned with an easy-to-use and yet universal means of knowledge representation - a conceptual graph processor with the technology and the interface that would make it usable by novice students. At present these provisions by the technology are purely speculative.

Conceptual modelling as practical epistemology

The confluence of the four str[illegible]s above with the educational effects enc[illegible]ed in commercial knowledge-engineering leads to a more general notion of conceptual modelling to describe the process.

Central to this idea is a view of the learner as a constructor of his/her own knowledge, as proposed by Papert (1980). It can be summed up in the proposition of *learner as epistemologist*. It captures an important abstraction from learning, which broadens and distinguishes it from a matter of communication and of memorization. Conceptual modelling in an interactive and reconstructable medium is a concrete expression of the proposition. That is to say, conceptual modelling directed by the learner is *practical epistemology*.

This view helps to meet an objection, put forward by Yazdani and Lawler (1986) that a "model-based epistemology of instruction does not exist". Concept model-based systems teach very concrete and specific things about knowledge. Its real "instructional strategy" is to demonstrate, through a student's modelling activity, a number of personally and socially significant epistemological statements. The ability to *run* a concept model is an instructive and live demonstration of the answer to the question one frequently ask of other people: how come they can think (or say) that? This is *practical epistemology*.

Conclusion

The useful fate of Prolog in education is to open and join the way to educational knowledge-engineering, within a general framework of conceptual modelling. Together with work in wide and sometimes apparently disconnected areas, it contributes to the possibility of a new approach to teaching and learning that incorporates many of the strengths and ameliorates some of the weaknesses of its precursors.

Bibliography

Adams, T. (1986) *Towards a Theory of Microworlds*, in Salvas & Dowling (1986), 312-320.

Ausubel, D. P Novak, J. D. & Hanesian, H. (1980) *Educational Psychology: a cognitive view*, 2nd ed., Rinehart and Winston, New York, 1980.

Briggs, J., Dean, J. & Nichol, J. (1985) *Toolkits for Naive Users of Prolog*, Proceedings of the AISB Artificial Intelligence and Education conference, Exter, 1985.

Conlon, T. (1985) *Start Problem-solving with Prolog*, Addison-Wesley, 1985.

Cumming, G. & Abbott, E. (1986) Prolog and Expert Systems in Education, *PEG-BOARD 1* (2), Winter 1986, and in Proceedings of the First Australian Artificial Intelligence Congress, Melbourne, November 1986.

di Sessa, A. (1982) *A Principled Design for an Integrated Computational Environment*, Laboratory for Computer Science, MIT, Technical Report MIT/LCS/TM-223, July 1982.

Ennals, J. R. (1984) *Teaching Logic as a Computer Language in Schools*, in Yazdani (1984), 164.

Ennals. R. J. (1985) *Artificial Intelligence: applications to logical reasoning and historical research*, Ellis Horwood, Chichester, 1985.

Fikes, R. & Kehler, T. (1985) The Role of Frame-based Representation in Reasoning, *CACM* 28(9), 904-920.

Friedland, P. (1985) Introduction to Special Section on Architectures for Knowledge-based Systems, *CACM* 28(9), 902-903.

Genesereth, M. R. & Ginsberg, M. L. (1985) *Logic Programming, CACM* 28(9), 933-941.

Gray, M. (1984) *POP-11 for Everyone*, in Yazdani (1984), 252-271.

Hayes-Roth, F. (1985) Rule-Based Systems, *CACM* 28(9), 921-932.

Kay, J. (1986) *Interactive Student Modelling using Concept Mapping*, Proceedings of the First Australian Artificial Intelligence Congress, Melbourne, November 1986.

Kowalski, R. (1984) *Logic as a Computer Language for Children*, in Yazdani (1984), 121-144.

Nichol, J. (1986) "When does a bandwagon become a hearse...", *PEGBOARD* 1(2), Winter 1986, 5-6.

Novak, J. D. & Gowin, D. B. (1984) *Learning How to Learn*, Cambridge University Press, 1984.

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas*, Harvester, 1980.

Salvas, A. D. & Dowling, C. (1986) *Computers in Education: On the Crest of a Wave*, Proceedings of the 1986 Australian Computer Education Conference, Computer Education Group of Victoria, Melbourne August 1986.

Sloman, A., Hardy, S. & Gibson, J. (1983) POPLOG: A Multilanguage Program Development Environment, *Information Technology: Research and Development*, 2, 109-122.

Sowa, J. F. (1984) *Conceptual Graphs: information processing in mind and machine*, Addison-Wesley, 1984.

Stern, N. R. (1986a) *Prolog as a Mind Tool?*, Proceedings of the First Australian Artificial Intelligence Congress, Melbourne, November 1986.

Stern, N. R. (1986b) *Prolog and Education*, Proceedings of CALITE 86, Adelaide, December 1986.

Yazdani, M. (1984) (ed) *New Horizons in Educational Computing*, Ellis Horwood, Chichester, 1984.

Yazdani, M. & Lawler, R. (1986) Artificial Intelligence and Education: an overview, *PEGBOARD*, 1(1), Exeter, Spring 1986, 68-74.

Increasing active learning in a CAL environment

William A. Stewart and Jack Flanagan,
Nepean College of Advanced Education,

Computer are claimed to have a powerful role to play in tertiary education, yet most programs available in the business education field have not displayed a quality which makes the claims for computers convincing. Beyond specific-purpose computational aids, progress has been slow. At the heart of this slow progress is the demand on the computer- learning innovator to cover three disparate fields: the academic skills for her/his discipline, the educational skills for effective communication, and the systems and programming skills to achieve an understanding of possibilities for CAL.

This paper uses one element of a managerial finance course to explore these issues as they were experienced by the presenter and his co-author. By demonstration the paper traces the origin of a program, its development in early use, and program modifications as the experience of four semesters' use demonstrated that learning experiences fell short of the authors' goals. Modifications were continually attempted to improve the active learning aspect. The paper presents a preliminary evaluation of the learning outcomes and generalises these to similar tasks in the business education environment.

A feature of accounting and financial software is that the system controls of accounting programs are often a direct hindrance to the new user. The penalties for wrong entries are usually re- entry of data in separate runs, enshrining all mistakes as a permanent reminder of misdemeanour. A potential benefit of computer learning has always been the non-threatening nature of mistake correction. Mistakes are electronically erased, and only finished successes show: the accounting security and audit trails are thus in direct conflict with educational goals.

Problems for business education software

There are substantial differences between business software and educational software on business topics. The focus for business software is on input-output relationships where the outputs from the system are the major benefit. Most business software on any specific topic will use a "black box" approach, hiding the processing aspects from the user.

Educational software on business topics has a different set of needs. While the outputs, the results of data manipulation or computation, are useful, the educational aspects of most interest are usually the relationships, and the processes themselves. The assumptions that have been made about the environment are also of importance. All of these aspects the computer must make available to the student. For such reasons many business educational institutions have taken up the spreadsheet as a contribution to business education.

Of course the spreadsheet is also a widely used tool in business, and knowledge of such a tool must enhance employability, but the educational aspect is that the student must develop the model her or himself, and thus focus on the process, not solely the outputs.

For the one-page process such as a budget or cost analysis the learning time for the spreadsheet is reasonable and the ability to manipulate data on a "what-if" basis can provide powerful insights. Our purpose in this project, one in a series of attempts to use the computer in business education, was to provide some of these aspects, but as well to develop something more elaborate as a teaching tool.

Problems from developing your own software

As a general comment, much of the software received in business education to date is poor, particularly by the standards of commercial products. An exception to that assertion is the range of accounting systems, where the available products in both business and education vie with one another in inscrutability, unfriendliness and just plain "orneriness".

There are reasons why good education software for business courses is slow to arrive, for three skills, not the usual two (knowledge of problem/knowledge about systems-programming), are demanded. The software developer must have a good technical understanding of the area, must have strong educational motivations, goals and insights, and lastly must acquire some system and programming skills to execute the technical and educational tasks to meet the objectives of the system. We might add also that there are few if any financial rewards for such developments.

Beyond that, the prescriptions for developing appropriate software at the tertiary level are difficult to find.

In a previous working paper (Flanagan/ Stewart 1985) we explored ideas from the education area which might guide the preparation of business education software. Of particular interest were the works of Gagne(1965) and Ausubel(1968) on the cognitive processes of education. In addition, Rushby(1975) had written on the computer-specific applications of educational ideas. (For brevity of this paper, we have chosen to provide discusion on these points by way of a handout to workshop participants.)

What emerged from this study was a set of general guidelines. We did not have a position on whther our students were holistic or serialistic learners. It was in keeping with the method of teaching that the process of the progrma followed a serialistic pattern, bue important that the student should have the widest possible control over the sequencing of access to aspects of the program.

User-friendliness was a vogue term not given much "operability" in the literature we read, but in our efforts was to be represented by careful control of inputs to avoid program crashes, context-sensitive help, and simple recovery processes from errors. The setting in which the user operated was the crowded typical college/university computer room, and use of bell or beep, a humiliating announcement to the room that the user had "goofed" again, was to be avoided.

Where possible, we would try to have educational software mimic the better efforts of commercial software to provide a familiar interface, and one that operated better than the restrictive and censorious software we were currently using.

As proceeding in such poorly charted territory represented such a threat, it seemed appropriate to tackle small tasks with apparently simple requirements. Specific feedback on success or failure could be ascertained more readily if the task were small and specific.

Our purpose in this workshop is therefore to explore our experiences so far, and by such exposure help others attempting to cover the same territory. We hope to have commentary giving other explanations as to why/where we went wrong, or where we should have gone instead.

This particular task

The subject chosen for exploration was the financial evaluation for the comparison of leasing with other financing options, sometimes shortened to "lease versus borrow-and-buy". As an educational task it involves a series of computations, present value analysis, and the understanding of a set of environmental relationships which affect the financial outcome. A simple 5-year example involves about 70 sequential computations.

Because of the computation volume, students, in our teaching experience, are loath to do much exploring of possible combinations, and gain insights to the interactions and influences which determine the decision outcome. Lecturers are none too keen either to create and mark examples, and the texts can usually afford one simple-to-medium complexity illustration. Rarely are there enough illustrations for practice, nor sensitivity-tested parallel examples. Illustrations in class are excessively time-consuming unless trivial examples are chosen and demonstrated at speed.

The program goals

The program goals were to :-
- make available a demonstration program which solved the lease versus borrow-and-buy analysis
- permit the student to generate problems for self study with answers to permit self-assessment
- tutor and explain the problem and process sufficiently to permit students to have their questions answered without lecturer assistance
- provide references for further reading
- provide a browsing facility to stimulate curiousity.

Operationally, the program would require:
- no instruction beyond disk loading and program name
- crash-proof data entry
- non-threatening error correction
- no consumables beyond hard copy
- minimized likelihood of assistance requests from data processing personnel
- outputs in a familiar or expected format.

Program failure and the first revision

The first implementation of the program ran in compiled Microsoft Basic on DEC Rainbows. It had been tested on a student who had an unreasoning fear of anything involving mathematics. As an assignment, she was asked to use the program to help her with the task, report on difficulties or weaknesses, and suggest improvements. She was tested for competence at the conclusion of her assignment.

As a result of her suggestions, before some processes could be accessed, a multiple-choice question which addressed the major purpose of the process was installed to act as a "sentry" to each process. Before a process could be used, the question had to be answered correctly. A short tutorial on what was wrong with an answer followed an incorrect response.

Confident in the usefulness of the enhanced program, the program was used for several classes the following semester. Apart from some minor problems with data entry, students succeeded in using the program for a conventional assignment. Subsequent testing of their ability to do the same type of problem manually showed no sign that the program had assisted them to gain the intended skills or insights. Post-mortem discussions with students showed that few had been curious enough to use the help screens or the tutorial, and had also failed to absorb the understanding of the broad processes guarded by the multiple choice questions.

The second revision

For the following semester, a more lengthy assignment required the students to do a variety of problems using the program; a series calculating tasks which would have been too onerous manually. It was thought that more continuous exposure to the program might have stimulated experimentation and enquiry with the use of the tutorial and help screens.

Tutorial discussions, class testing and oral quizzes revealed that more use had been made of the program features, but that too large a proportion had asked other students for the answers or rote- learned the required responses and processed the information mechanically without learning taking place.

It is reasonable at this point to divert to question whether we had arrived at any more than a reminder of the adage : "you can lead a horse to water but you can't make it drink" and therefore that you cannot force learning to take place. There is truth in that, but it must be understood also that the student is under competing pressures to acquire all sorts of skills and knowledge. Knowing when one has the required knowledge and skills so that it is safe to switch priorities, is one of the difficulties of a content-overladen curriculum.

We decided therefore that the signals on when it was safe to stop needed to be clearer to the student. The structured set of problems was extended in the following semester. These required resetting of the problems, particularly the environmental features, and needed closer study of the computations if the questions were to be answered.

The results of this stage showed that students had made more use of the help facilities and had used the options in the program more than previously. Some students suggested further computations and presentations which would have made data available in forms that matched their way of thinking about the problems. While this represented progress of the sort we had hoped for, there was still a disappointing lack of competence in performing the actual task of an analysis under exam conditions. As a generalisation, it could be said that their interest and understanding had been increased, but that the performance of particular skills had not improved to the degree hoped for. A series of intensive workshops would, on past experience, have brought the students to a higher *skill* level.

The present program

As the inability to display the technical skills of the analysis seemed the problem, the present revision requires the student to complete one iteration of each of the key analyses. The program makes the remainder of the analysis available after an attempt. Wrong answers are flagged, but the analysis is always provided.

After two successful attempts at each of the tasks, the student is freed from further demands by the program, and may make any number of further changes to inputs without entering one line of the iteration.

Class testing indicates that this procedure has had some success in improving this level of skills, though at the time of writing (July 1987) there has not been a testing under final examination conditions.

Summary

Our purpose in this workshop has been to discuss the progress of an attempt at the development of a computer-assisted learning process. The goals of this process can be seen in hindsight as naive in that the program provided an opportunity for experimentation and serendipitous discovery. I.

the task- intensive environment of a business school the stresses on the student do not provide the atmosphere for such a learning process for most students.

Some level of success has been reached by a more exacting setting of the problems which the student has been required to explore or solve. However, the program became again a tool to assist lea ning rather than a self-contained environment of learning, which had been the unrealistic goal.

Because many business problems require both an understanding of the setting and the problems within that setting, with also a set of exacting technical skills for problem solving, others may be forced to the stratagems we have employed. The powerful computational processes which the computer can carry out and display may pass before the eyes of the passive student without any response. Forcing the student to participate in an iteration of the solution, as well as problem solving for answers on the influence of environmental factors, seem to have been responsible for the improvements we have noted.

Bibliography

Ausubel, D.P., (1968), *Educational Psychology: A Cognitive View.* New York: Holt, Rinehart and Winston, Inc.

Gagne, Robert M., (1965), *The Conditions of Learning,* New York: Holt, Rinehart and Winston. Inc.

Applying intelligent CAL principles to help subsystems for interactive software

David A. Waugh, Department of Computer Science
Kevin P. Stark, Department of Civil and Systems Engineering
James Cook University

A multi-level dynamic Help subsystem for interactive software is described. By combining information about the user's history and present situation a system can offer help that is directly related to a user's needs. A number of help techniques are discussed including dynamic example generation.

There are many similarities between offering intelligent help to a user of an interactive software package and intelligent tutoring in computer assisted learning (CAL) systems. In both environments, an accurate model of the student, or user, must be maintained if appropriate assistance is to be offered. A help system, however, is not concerned with teaching concepts involved with learning how to solve problems with a computer, or presenting lessons to the user to be answered and evaluated. Intelligent help systems need to offer more than static script based assistance driven by some form of lookup mechanism. To be truly useful, help must be personalized. Personalized advice implies keeping a model of the user and analyzing this model to determine the appropriate form of coaching to provide. In this paper, we investigate the potential for intelligent help and present an implementation of a form of user modelling for interactive software with an intelligent help system.

Discussion

There are three main elements to consider when looking at providing an intelligent dynamic help subsystem. These are:

1. Hierarchical Help Systems
2. Computer Coaching
3. User Modelling

Hierarchical Help Systems

Most software today comes with some form of help system. These can be on-line or simply a set of manuals or some combination of both. Of the on-line help systems available, there is a wide range of assistance offered, from simple electronic copies of documentation to complete elaborate tutorial systems (Houghton, 1984). An example of the elaborate form of help is the Do What I Mean (DWIM) facility in the Interlisp environment (Teitleman & Masinter, 1981). This facility will attempt to diagnose and correct incorrect input while leaving it up to the user to accept or reject the correction. Obvious corrections are made without consulting the user. Other systems like the SIGMA message processing system have on-line tutorial assistance with built in lessons and exercises (Rothenberg, 1979).

Another consideration for help systems is whether an active or passive role should be taken. Fischer et al (Fischer et al, 1985) advocate passive help in a system for the subset of commands a user knows a little about and uses occasionally and for the set of commands the user thinks exist in the system. However, active help is required to introduce the user to the set of concepts and commands that exist in the system but the user has no idea about.

Passive help systems wait to be requested by the user. Typically, passive help is available in the form of a menu driven or keyword system. Menu driven systems have the advantage over keyword systems in that they are hierarchical in nature and display a set of keywords on which help is available. Natural language help systems exist but are still not sophisticated enough to provide adequate levels of help, primarily because of the problems inherent with a natural language interface. In general, passive help systems do not take into account the user's present or past situation (Fischer et al, 1985).

Active help systems are characterized by their ability to interrupt a user to provide some form of advice or coaching. Interruptions are not neccessarily restricted to correcting errors made by a user, but also offer a more efficient method of accomplishing a goal than the current method being used. An example of this latter form is the Unix advisor (Kay, 1986). The important aspect of active help systems is that they monitor the user's behaviour and some attempt is made to reason about their goals (Fischer et al, 1985). One major concern with active help is when and where to interrupt a user. A help system that continually interferes becomes a nuisance whereas an active help system that never offers help is simply a passive help system. Some form of judgement needs to be used to adequately service the user's needs. Support should stop when a command has been successfully mastered by the user. One method is to give the user control over the level of help interface they are using.

Computer Coaching

Personal tutoring is a very effective method of teaching, especially when coupled with additional formal instruction either individually or in a group. It is important that a tutor does not interrupt too frequently as this can lead students not to discover from analyzing and attempting to correct their own errors. A tutor must know when to interrupt a student and what to say when this interruption occurs (Burton & Brown, 1982). These principles apply to designing an effective help system. In a help system the advice giving element is basically a computer coach. To offer the most effective advice, a tutor must know something about how students learn and when they are the most receptive to such advice. The same applies to help systems, except that a user often initiates the interaction. With a dynamic help system the 'coach' takes on the role of tutor by not waiting for the user to ask for help. But, the scope of assistance that a help system provides is limited to the concepts and commands of the interactive software that is being used. A tutor, on the other hand, offers advice on all aspects of the problem at hand, not just the tool currently being used. This last point means that an intelligent dynamic help system needs to embody much less information in its knowledge base. This also implies that less knowledge needs to be stored when compiling a user model.

User Modelling

A student model has been defined by Clancey, (Clancey, 1986), as a set of records that describe a student's knowledge about what is being taught so that the program, with the interaction of the teacher, if necessary, can adapt its presentations to the student's needs. The classification of student or user models can be achieved using the following major characteristics (Rich, 1979):

- the model is of a canonical user or the models are of individual users
- construction is explicit by the user or abstracted by the system from observing users' behaviour
- the model embodies short-term specific information or long-term general information.

For a user model to be useful it must continually evolve to accurately reflect the change in a user's knowledge. This can be achieved by maintaining a record of a user's correct and incorrect usage of a system. According to the classification above, the model described in this paper acquires its knowledge implicitly to create an individual model of a user. The kind of knc edge that is embodied is primarily short-term knowledge about specific constructs and concepts. However, as a user builds knowledge over time, a long-term model is available by continuous analysis of the acquired data. At some point, a user could be determined to have mastered a system and the form of advice offered would automatically change to a passive role.

There are various methods of implementing a model. Some of these include overlays using genetic graphs (Goldstein, 1981), stereotypes (Rich, 1979), scripts (Schank, 1977), frames (Minsky, 1975), schemas (Bobrow and Norman, 1975) and specifically for CAI using a procedural student model (Self, 1974). The model discussed in this paper uses a form of frame-based representation.

The User Model

The only way to build a help system that is not continually interrupting is to build and maintain a model of the current user. This model would represent a profile of a particular user's history of usage with a particular interactive software package. This model would need to embody the various attempts (correct and incorrect) made by a user when interacting with a system. The obtrusiveness of the help system can then be controlled by scanning the user's model to determine if help should be offered for the current situation.

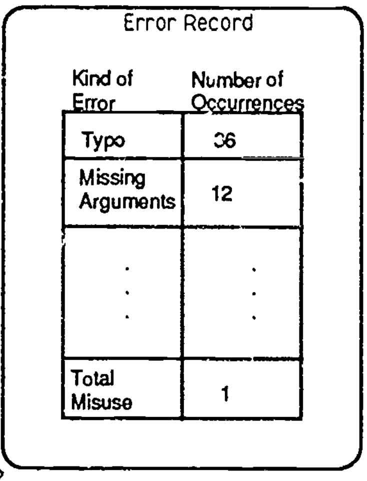
To employ an effective user model, we need to consider the following:

- The kind of knowledge we need to model.
- The method of implementing this model.
- The application of this knowledge by the help system.

The kind of knowledge we need to model.

The kind of knowledge we need to model includes the number of times a user has correctly and incorrectly used each of the commands and/or concepts available in a system. Incorrect usages need to be categorized in terms of the determined severity of the error, ranging from a typographical error or misspelling to a complete lack of knowledge of the correct form of a command or concept. Each command may have a list of possible errors and the number of occurrences of each kind can be compiled as in fig 1. Also, to aid the help system in determining its effectiveness, for each command a record of the number of times help has been offered and what strategy was used should be kept. A later analysis may show some forms of help to be less effective than others for some commands/concepts. A further analysis of many user models could indicate some common areas of trouble or misunderstanding. These last two points are very attractive to educators using software with such a help system built in.

Figure 1



Error Record

Kind of Error	Number of Occurrences
Typo	36
Missing Arguments	12
. . .	. . .
Total Misuse	1

The method of implementing this model

The model is implemented in a hierarchy of frames. The top level frame represents the knowledge about all users of the system, (see fig. 2). It includes the number of users for which a model is kept, a current list of the ten most common errors in the system and one model for each individual user. The model for each individual user, (see fig. 3), includes some form of user identification, a distinction between the number of spelling errors and construct errors and a collection of usage records for each command/concept that user has used. This collection of usage records grows as the user attempts new commands/concepts in the system. The omission of a record for a particular construct implies the user either does not know of its existence or simply has not had a need for its use yet. Each usage record,(see fig. 4), includes which construct/concept is being recorded, the number of correct usages, the number of incorrect usages, the number of times help has been offered, the number of times a user has asked for help for this construct, which help strategies have been used and a compilation of the kinds of errors encountered as in fig. 1, and a link to the set of static library examples provided for this construct by the system.

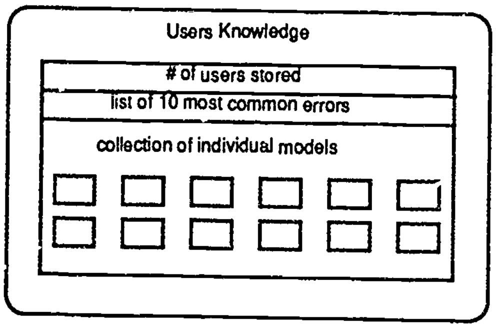

Users Knowledge
# of users stored
list of 10 most common errors
collection of individual models

Figure 2

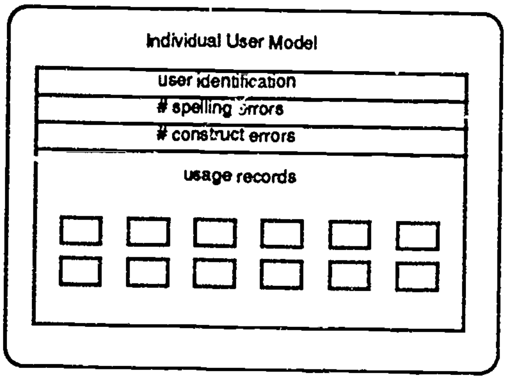
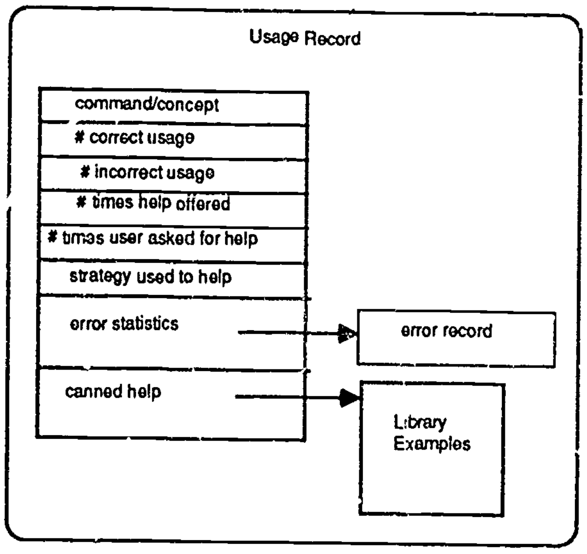


Figure 3

Figure 4



The application of this knowledge by the help system

To apply the knowledge embodied in the model, the help system evaluates the user model to determine both the appropriate level of help and the specific example or suggestion to provide to the user. For example, if the user has used a command incorrectly the help system will check first to see the number of times a user has correctly used this command. If no record exists, then this is the initial usage by the user and a dynamic example could be presented to the user. A dynamic example is a powerful way of illustrating to the user the correct form of their specific input through the use of an example. If a record does exist, the confidence factor is calculated. The confidence factor for a command or concept is the ratio of correct usages to incorrect usages determined for the types of errors a record is kept. If the confidence factor is high, say at least 3:1 and the kind of error is a spelling error, or minor omission, then the user is determined to have mastered this command and the form of help may be to automatically correct the erroneous input while asking for the user's approval. If the form of the error is determined to be major, then a library example would probably be an appropriate form of help.

One problem arises when a user is determined to have mastered a command or concept and an erroneous usage occurs. If this error is merely a spelling error, the help system treats this as noise in the model. However, the help system has no method of handling the situation where a friend of the user for whom the model has been created uses the system. Suddenly, we have a completely different pattern of usage evolving that contradicts any assumptions made using the existing model. The only way to handle this currently is to strongly discourage the above practice and if this situation is detected by the help system by encountering many errors in a previously mastered command, the system can ask the user if they are really who they should be. One possible honest cause of this situation would be a user who has mastered the system then not used the system for a prolonged period of time and returned to use it again. One could argue that the returning user really is not the modelled user from before. To compensate for this, the model could include time stamping and erroneous usage could be checked against the previous date to detect any 'staleness' about the user and this command.

To determine adequate levels of help and appropriate examples requires a substantial amount of time and effort. This is analogous to a tutor who must determine the correct form of help to provide to a student. No two students are the same, hence an effort must be made to personalize help and tuition. Of course, when a student masters a system, help becomes passive and inexpensive to provide.

Domain of Application

Novice programming environments (NPE) seem to fall roughly half way between a software engineering tool and a CAL system for teaching programming. Users are trying to develop software to solve some specified problem but the reason for solving this problem using a computer and a programming language is to learn how to program in that language (in addition to learning other problem solving concepts). A useful programming environment (PE) must thus provide the user with some tools to enhance the efficiency of the programming task. To deal effectively with novice users, some consideration must be given to accommodate the fact that the user is learning by doing rather than an expert purely trying to write a program. This consideration can be in the form of a useful help system that changes dynamically with the user's anticipated needs. One method of accomplishing this is to model the user's history of the various language constructs and concepts to use as a reference when offering help.

A NPE for Pascal programmers using Macintosh computers is currently under development at James Cook U[illegible]y, (see Waugh, 1987). The current version of the system uses a static help system that is obtrusive and predictable. A user builds a Pascal program statement by statement. When an error is encountered by the system, an attempt is made to generate a suggestion which is then offered to the user to

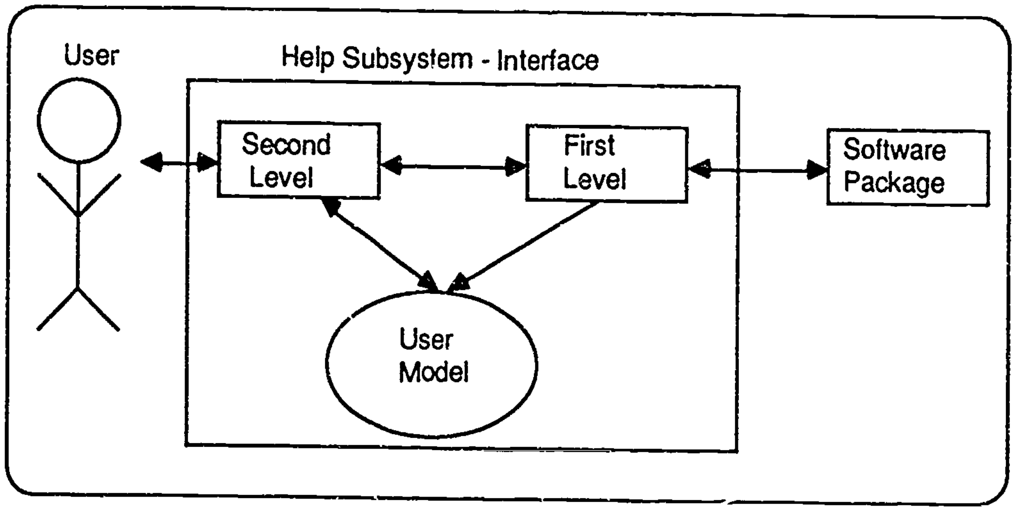


Figure 5

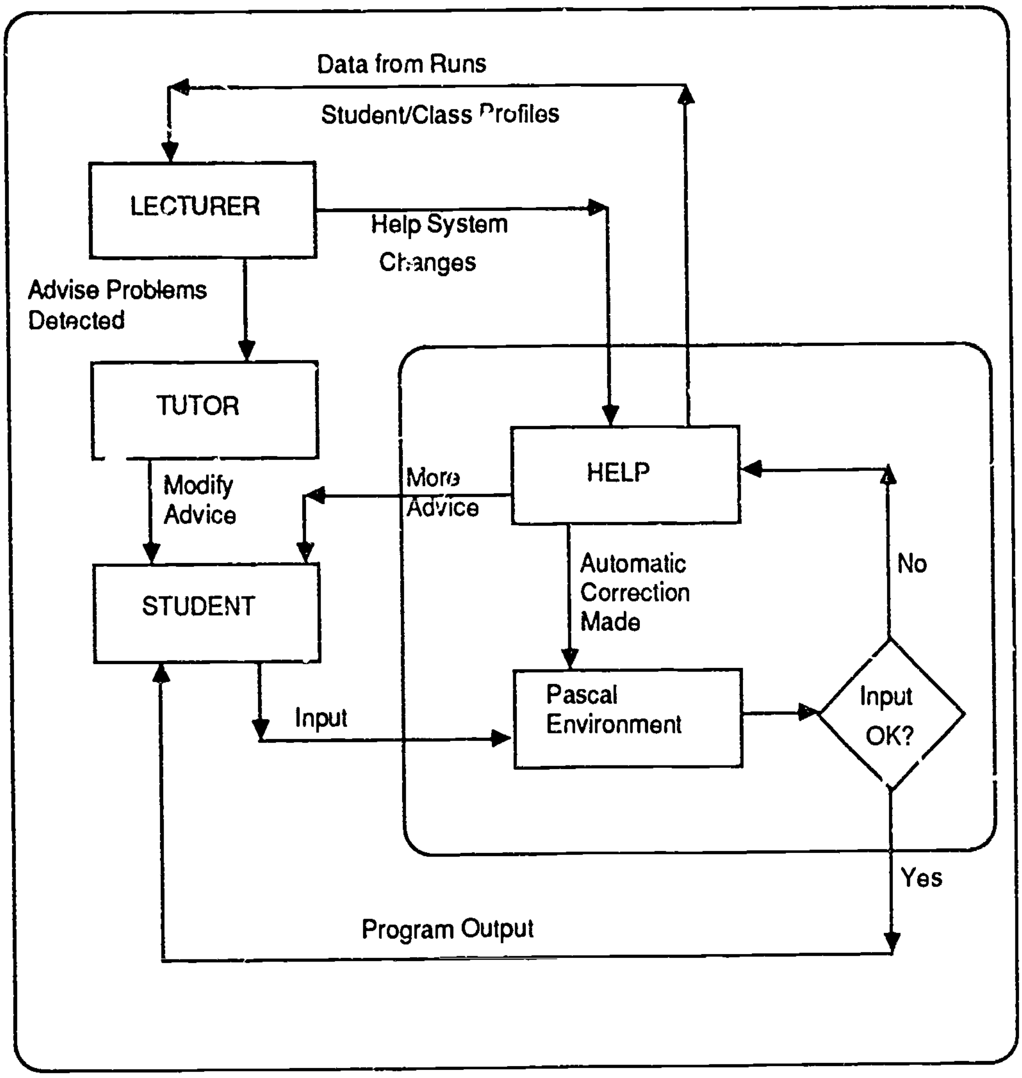
accept or reject. If a suggestion cannot be generated, the user is told the current statement is unrecognized by the system. An indication as to where the error occurred together with an explanation of the type of error is provided (as per typical compiler output). A static help system is always available in the form of a menu-driven keyword based system. At any time a user may browse the help system for an explanation and/or example of any concept or construct for which help is available.

This form of help can be improved by providing a multi-level dynamic help subsystem as described in this paper. The full potential of these ideas is realized by giving the user some control over the level of interface they are interacting with. The help system is organized into two levels (see fig. 5), the first being for experienced users while the second is for novice and inexperienced users. The first level provides a script based system coupled with the ability to generate a dynamic example using the user's erroneous input should the user request it. The second level provides the same functionality as the first only the user model is consulted on every erroneous input to determine if advice should be given. All users have the option of choosing which level of help they would like, but many novice users would usually choose the interface that would be most beneficial to them. This allows an experienced user to avoid the overhead in the system of building and maintaining the user model as well as avoiding the dialogue and interruptions inherent in the intelligent help level. The first level interface maintains information about the user by modifying the particular user model (hence the one way connection in fig. 5) in case the user reverts to the second level interface during a later session.

The overall implementation of such a help system incorporates the lecturer, tutors and students for a particular course and the help subsystem for a particular package, introductory Pascal programming and a Pascal PE in this case. The interrelationships of these various elements are indicated in figure 6. Thus if a particular student continues to make a particular error, the level of HELP provided is increased and if a class of students persistently make the same error, the HELP facility detects a teaching problem and advises the LECTURER to check and modify, if appropriate, the tutoring approach for that class, both human and HELP, as indicated by the connection from HELP to LECTURER labelled 'Data from Runs' and the corresponding links be'ween the LECTURER and HELP and TUTORS. In each session, a student enters INPUT to the system, (primarily Pascal statements), which are incrementally compiled by the Pascal environment. Upon detecting an error, the HELP system is invoked which offers advice to the student depending on the evaluation of the student model. If no error is detected, the student is allowed to continue uninterrupted.

The development of this student model therefore involves elements of artificial intelligence with cybernetic feedback loops and a heuristic learning facility combined with elements of database management with accumulated statistics for individuals and classes of users. After a number of uses of the HELP facility, assessment of each student's ability in Pascal can be determined in structural terms both semantically and syntactically and by aggregation the whole class can be rated. The HELP system has been developed heuristically

Figure 6



with an interactive learning component - thus

- the first HELP procedures are determined using our experience as a lecturer of Pascal and introductory programming over the past 3 years.
- a record of the last 10 errors of each type is maintained continually for access by the teacher. This allows modification of HELP procedures, introduction of new prompts and the hierarchical ordering of the search for errors used by HELP.

It is important to realize that the intelligence level of the student as a Pascal user will normally increase at a rapid rate and the need for the HELP facility should correspondingly quickly decline. The better the HELP system is, the lower the number of intrusions required by the HELP facility will be.

Conclusions.

In this paper we have shown the parallel between tutoring and on-line help systems and presented a method for tailoring the help to a specific user with the inclusion of a user model. Script based CAI models only capture the statistics of the number of incorrect matches of the user's answer against the set of expected answers.The method presented overcomes this restriction of script based CAI models by the inherent embodyment of the system expert in the help system and subsequently the user's model.

It should be noted that we do not advocate the use of such intelligent help systems to replace the role of human tutors, but rather to enhance the environment in which students and tutors interact. Such a system would take care of many minor problems, leaving tutors with more time to handle the substantial problems like the classic 'where do I start?' situation which is only just beginning to be attempted to be automated along with other logic error detection problems. A potential problem with intelligent help systems is where the machine incorrectly perceives the user's ability forcing the user to do something they did not intend to do. This can be solved by giving the user control at all times to accept or reject any action taken by the system. Also, the help system should be as transparent as possible to the user so that responses are brief and succinct so as to minimize the delay and disruption in a user's thinking and acting cycle.

This form of help also provides the useful feature of individual as well as group feedback to the lecturer which promotes the continual growth and modification of the teaching style of the lecturer and tutoring styles of the tutors and help system. Such a dynamic situation can only enhance the efficiency and quality of the teaching process thereby providing room for enriched learning environments.

References.

Bobrow, D. G. & Norman, D. A. (1975). Some principles of memory schemata. In D.G. Bobrow & A. Collins (eds), *Representation and Understanding*. New York: Academic Press.

Burton, R. R., Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In (Sleeman & Brown, 1982), Ch.4.

Clancey, W.J. (1986). Qualitative Student Models, in Traub, J.F., Grosz, B.J., Lampson, B.W. and Nilsson, N.J. (Eds) *Annual Review of Computer Science*, Vol. 1. Palo Alto, California: Annual Reviews Inc.

Fischer, G., Lemke, A. and Schwab, T. (1985). Knowledge-based Help Systems. *Proceedings of the ACM CHI Conference.*

Goldstein, I.P. (1981). The genetic graph: a representation for the evolution of procedural knowledge. In (Sleeman & Brown, 1982), Ch.3.

Houghton, R.C. Jr. (1984). *Online Help System. A Conspectus. Communications of the ACM*, 27(2).

Kay, J. (198?). *Interactive student modelling using concept mapping.*

Minsky, M. (1975). A framework for representing knowledge. In P.H. Winston (ed.), *Psychology of Computer Vision.* New York: McGraw-Hill.

Rich, E. (1979). User Modelling via Stereotypes. *Cognitive Science* 3.

Rothenberg, J. (1979). On-line tutorials and documentation for the SIGMA message service. In Proceedings of AFIPS National Computer Conference, New York, vol. 48. AFIPS Press, Arlington, Va.

Schank, R. C. & Abelson, R. P. (1977). *Goals, Plans, Scripts and Understanding: An Enquiry into Human Knowledge Structures.* Lawrence Erlbaum Associates.

Self, J. A. (1974). Student Models in Computer-aided Instruction. *International Journal of Man-Machine Studies* 6.

Sleeman, D., Brown, J.S. (eds) (1982). *Intelligent Tutoring Systems.* London, Academic Press.

Teitleman, W. & Masinter, L. (1981). The Interlisp programming environment. *Computer* 14(4).

Waugh, D. A. (1987). MacTutor: An Intelligent Novice Programming Environment and Tutor. *Proceedings of the Fifth Annual Conference on Computer-aided Learning in Tertiary Education*, University of Sydney, Sydney.

David Waugh holds a BSc (Hons) Computer Science, 1985. The University of Western Ontario, London, Ontario. MSc in Computer Science, James Cook University, Townsville. Current Position is a Lecturer, Department of Computer Science, James Cook University. His research interests: Novice Programming Environments, Software Engineering Tools, User Modelling, User Interfaces

MacTutor: An intelligent novice programming environment and tutor

David A. Waugh
Department of Computer Science
James Cook University

The design and implementation of a novice programming environment with an intelligent tutor is described. The implementation presented is for use on Macintosh microcomputers by first year undergraduate computer science students learning Pascal. MacTutor provides an incremental compilation environment that detects syntactic and semantic errors on entry. A suggestion generator provides an 'intelligent' reconstruction of an erroneous statement and presents this to the user for acceptance. The environment includes full run-time simulation abilities for source code debugging with features such as single stepping, breakpoints and variable traces.

Introduction

Programmers face the challenging task of producing correct, efficient and reliable software. This process has been simplified by many programmer's development aids such as syntax directed editors, (Donzeau-Gouge et al, 1984; Morris & Schwartz, 1981; Marlin, 1986; Moretti & Lyons, 1986; Welsh et al, 1986), and by self contained programming environments (PE), examples include INTERLISP (Teitelman & Masinter, 1981), the Cornell Program Synthesizer (Teitelbaum & Reps, 1981), and Smalltalk-80 (Goldberg, 1983). However, there exists a subclass of programmers who depend greatly on human assistance to complete their task, these being novice programmers. The task of a novice programmer is twofold. One goal is to learn a programming language. The other is to produce coded solutions to various tasks set out by their instructor. Existing environments do not provide assistance in both areas. In fact, the assistance provided by programming environments to code a solution often imposes a significant learning overhead just to master the operation of the tool. This results in a vicious circle that often frustrates novice users enough to give up.

An ideal environment for novice users would provide useful feedback in terms of a creative suggestion as to why the error occurred and even perhaps a suggested correction. This kind of assistance can be developed for all of syntactic, semantic and run-time errors. As much effort as possible should be spent in reducing the overhead and effort in learning to use both the PE and the programming language. Another possible area of assistance available to the novice user is a help subsystem within the environment.

Such a programming environment is being developed and is presented here. The principles for this system are investigated and presented followed by the presentation of an environment for novice users learning the programming language Pascal on Macintosh computers. This environment is being developed for specific usage by a first year computer science class to solve programming assignments related to the course.

Some of the features to be provided by the environment include suggested corrections, syntactic and semantic error detection as each Pascal statement is entered, the ability to delay subprogram definitions until later, the ability to test and save subprograms individually and the automatic insertion of variables for declarations. Some of the run-time features available include single stepping through the source code, breakpoints, dynamic variable tracing, variable watchpoints that indicate each modification of a particular variable, plus the indication of any error that is about to occur like an array index out of bounds, or pointer referencing to nil. Static help is always available to the user in addition to the obtrusive help at the beginning. The system reduces the problem of response time degradation by using an incremental compiler.

The major advantage of this system is a great reduction of time spent by a novice user editing, compiling, executing, re-editing, recompiling etc. The learning overhead has been minimized by making full use of the menu and icon features available in the Macintosh toolbox. The appearance of the environment to the user was influenced in part by the environment presented by MacPascal, (Apple, 1985).

Discussion

Programming Environments

The term 'programming environment' can include systems that manage the complete software development process or an integrated tool that supports programming. In this paper the following definition for a Programming Environment given by Ardis et al, (Ardis et al, 1985), is adopted. This being: an integrated software system that directly supports the activity of programming. The systems that support the overall programming process are better labelled as software engineering environments.

A significant amount of work has been done developing environments for the programming language LISP. INTERLISP, (Teitelman & Masinter, 1981), is perhaps the most well known LISP environment that includes such features as automatic error correction, an integrated structure based editor, sophisticated debugging, a compiler, and a file system. This environment progressed from work by Teitelman, (Teitelman, 1969), where early ideas on programming environments were formed from observing experimental programming using LISP. Another significant contribution to the area of programming environments is the work done on the Cornell Program Synthesizer, (Teitelbaum & Reps, 1981). The Synthesizer views programs as a collection of computational structures and editing is restricted to structures defined by the underlying programming language grammar. The grammar is organized by templates for the structures and creation of a program is accomplished by selection and completion of these templates. The Synthesizer also provides a run-time environment with features such as single-stepping and tracing. Another PE based on a structured view of a program is the MENTOR-PASCAL environment, (Donzeau-Gouge et al, 1984). Editing in MENTOR-PASCAL is restricted to using a tree manipulation language, called MENTOL, to edit an abstract syntax tree. Limitations on the rigid nature of structure based editors are discussed by Welsh et al, (Welsh et al, 1986), who advocate that editing should conform to the user's conceptual form of a program and not impose what may be an unnatural representation on the user. Editing should be accomplished using whatever conceptual level a user is comfortable with which may vary at different stages of the program's development.

Incremental Compiling

A major problem with programming environments is the degradation in response time due to repeated compilation of the complete program for each modification.

Techniques to reduce this overhead include incremental compiling. The earliest work on incremental compiling was by Lock, (Lock, 1965), who describes a method of editing Algol 60 programs at the statement level. This allows for syntactic error detection and correction as each statement is entered. Semantic checking is left until run-time. The interactive Algol system CONA, (Atkinson & McGregor, 1978), was developed for use in an introductory programming course and involved complete recompilation of the source code. The authors reported that the resulting overhead imposed was quite acceptable to the novice users but not for sophisticated users. In an attempt to reduce the overhead, further work was undertaken to develop a form of incremental compiling using context sensitive editing with a structure editor, (Atkinson et al, 1981). A more flexible approach to incremental parsing allowing for modifications of any form is presented by Ghezzi and Mandrioli, (Ghezzi & Mandrioli, 1979). However, the flexibility gained in the editing unit is at the expense of the storage and maintenance of significant amounts of information. To avoid recompilation, the scope of each statement and variable must be maintained to determine the range of effect a modification will have.

Teaching novices programming

An excellent source of information concerning the various aspects involved with teaching novices programming can be found in the report by du Boulay and O'Shea, (du Boulay & O'Shea, 1980). The report is a comprehensive literature review on the topic up to 1980. Findings presented include ideas on system use like: not to underestimate the amount of effort a novice user may require to master a system, and statistics concerning the major sources of errors in a programming language, e.g. the most error-prone construct in Pascal was the semi-colon. Some reservations are expressed concerning the lack of significant progress made towards predictive theories about the programming process and that much of the data collected about novice programmers varies significantly depending on the language used and the environment in which testing was carried out. They conclude with:

> Bearing in mind these reservations, it is clear that all aspects of programming cause novices difficulty, including typing, planning, coding, testing and debugging. Many novices are frightened away right at the start by the amount of arbitrary detailed information that they have to remember, as well as the problems of typing accurately and quickly. Languages in which the novice can very quickly start writing and running interesting programs (e.g. LOGO) have a definite advantage in that early success breeds confidence. (du Boulay & O'Shea, 1980).

These principles can be applied to many other categories of users, not just novices.

Suggestion Generation

Work on suggestion generation began with early work on error-correcting compilers, (Aho & Peterson, 1972; Graham & Rhodes, 1975 ; Kantorowitz & Laor, 1986; Moura, 1986; Feycock & Lazarus, 1976; James & Partridge, 1973). The techniques described in this paper most closely resemble ideas from James & Partridge. They represent a method of representing the terminal symbols of a grammar as a tree and perform an attempted match of the input statement's tree to the correct form's tree. One interesting technique also presented is the confidence jump that allows the parser to 'jump to a conclusion' based on a partial match to a certain degree. This could lead to an erroneous intention diagnosis and if returned to the user as a suggestion, it could be misleading. However, erroneous intention diagnosis and resulting suggestion can often indicate the source of the error to the user.

Requirements Specification

The following requirements were identified and specified before a design was initiated.

a. Syntactic and semantic check as the code is entered.
b. Suggestions provided when the system can anticipate the user's intent.
c. Force corrections to be made as the errors occur.
d. A file system to save work in progress as well as finished programs.
e. The ability to save procedures and functions separately to build a library of routines for later use.
f. To allow the delay of procedure and function declarations.
g. Offer dynamic help with examples and rules.
h. Menu driven user interface.
i. Use independent windows for display of input/output.
j. The ability to trace code and variables during 'simulated' execution using breakpoints and watchpoints.
k. Automatic insertion of variable declarations.
l. Automatic indentation at various block levels.
m. The ability to test a procedure or function independently.
n. Detect run-time errors during simulated execution.

The user is assumed to be a novice programmer learning the programming language Pascal. No previous programming experience is assumed. It would help if the user has had a single tutorial on using the Macintosh prior to their first session with the system.

Design Principles

The system was designed using top-down principles combined with the advantages from prototyping and successive versions. At all times we maintain a system that runs and is therefore testable while only offering a limited subset of the projected capabilities. This makes the project continually exciting to the developers because we always have something real to work with, instead of a simulation of 'what it is going to be like'. Successive versions have the added advantage of continually testing the existing software and design principles under new circumstances. This helps to remove many otherwise unobtrusive errors. Also, we managed to discover how the user interface principles and assumptions worked out over a long period of time (1 yr.). An idea for a response mechanism, like mouse click, may sound like a good idea at the start, but repeated use can illuminate the frustration that can occur.

The basic system was broken down into the following modules. (see fig.1), User Interface, Editor, Command Processor, Runtime Controller, Lex/Parse and Encoder/Decoder. This breakdown minimized the area of effect of the Macintosh specific features, the toolbox routines in particular. These machine dependent peculiarities were confined to the Editor and User Interface modules, a technique known as information hiding, (Parnas, 1972). Information hiding permits the independent development of modules, a useful software engineering technique, while also encapsulating details in one area thereby minimizing the effects of changes in other modules. Using this technique and porting the software to another machine implies that modifications will be restricted to the module containing the machine dependent details.

The Editor and User Interface modules were developed first resulting in a working editor. We then used this editor in our first year class as a simple editor for entering programs into MacPascal. This gave us onsite testing by the actual users of the system enabling us to compile feedback on design

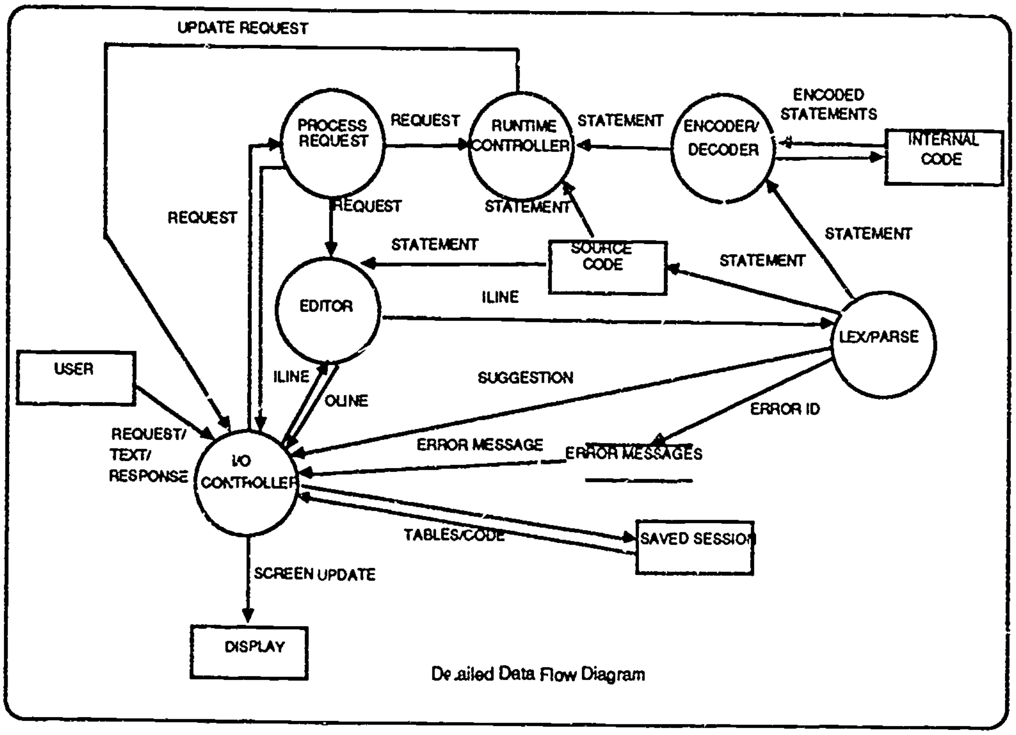


Figure 1

issues much earlier than if we had waited to release the final version of the system. The next module under development is Lex/ Parse. This involves the lexical analysis and parsing of the Pascal source statements entered by the user. The lexical analyser was originally developed using Lex, (Lesk & Schmidt, 1975), under Unix by specifying a set of regular expressions to describe the syntactical elements of Pascal. The output from Lex was a working lexical analyzer which was then modified to work on the Macintosh under Lightspeed C and our environment. This method of development proved to be useful due to the increased speed and versatility of Unix on a Microvax II with spare capacity, however, a unified development environment has some obvious advantages. All development would have taken place on the Macintosh if Unix had been available.

The development of the Lex/Parse module is incremental, implementing one grammatical category at a time. This allows for continual testing and the maintaining of a working environment at all times. Work on the Suggestion Generator module is proceeding in parallel with the Lex/Parse module. This is facilitated by the strong relationship between the original parse and the method of generating a suggestion as discussed later.

Description of the major features of the environment

Immediate detection of syntactic and semantic errors:

A user enters a series of Pascal source statements into the system to create a program. As each statement is entered, it is parsed by the incremental compiler to determine if the statement is constructed in a valid form. This will detect any syntactic, and some semantic errors present in the statement. If any errors are detected, the user is informed and must correct them before being allowed to continue. This ensures that the code entered so far, is error free as far as the parser is concerned. If the system can anticipate the user's intent from an erroneous statement, a suggestion is generated and presented to the user for acceptance or rejection. If the user deletes any code from the program so far, the system will check this deletion for any errors introduced into the remaining code. Again, the user is notified and suggestions are generated if possible.

Procedure/function testing and saving:

Students taking an introductory course in computer science are often taught to develop their software in a top-down manner. This involves the continual refinement of modules and routines. The Pascal programming language is often used to teach introductory programming skills, however, Pascal requires that all routines be declared before their usage. This requires bottom up implementation for a piece of software that was usually conceived in a top down fashion. This anomaly is often the source of confusion in many novice programmers. They are required to worry about all the details before getting the overall process correct. One alternative is to use stubs, merely a place holder, for routines at a lower level, returning to define them in detail later. The MacTutor environment provides another alternative, this being the ability to use a routine call without defining the body of the routine first. A list of routines that need to be declared before a program is complete is kept. The user may test routines individually as well. A test shell is provided to specify the parameters required for a routine and then simulate the execution of the routine. This feature allows the user to test using a bottom up, top down approach or both, depending on the user's preference. This provides a means to allow the user to develop their programs in a form that directly corresponds to their personal conceptual form. The last feature is the ability to save routines in a library for later re-use. This feature should be brought into context though, in that Pascal routines are very difficult to use in two different environments due to the strong typing in the language but this does not make re-use impossible.

Simulating execution:

An extremely effective method for a user to discover errors is to actually run their program. The problem with conventional compiler environments is that a run is accomplished independently from the editing environment. Any errors that occur must be corrected by leaving the run time environment and entering the editing environment. The cycle of edit - compile - run is repeated until a correct program is achieved. One advantage of a self contained development environment is the ability to switch from one function to another immediately. Another advantage is to be able to control the execution of the code segment and stop and examine variable contents at the user's discretion. MacTutor includes the ability for a user to define where they would like a simulated execution to suspend and wait for further commands through the use of breakpoints and watchpoints. A breakpoint is associated with a source code statement. When the run time controller encounters this statement, execution is suspended until the user chooses to continue. A watchpoint causes execution to pause when a certain variable is modified. Another function is the ability to trace certain variables throughout the program execution. The contents of the specified variables are displayed in a window at all times with the appropriate modifications being made as they occur. The user also has the ability to step the run time controller through the source code one statement at a time using the single stepping function. These tools give the user a very effective method to trace the program's execution under various conditions and inputs. Program debugging becomes an enjoyable and informative task while also being efficient. Using these techniques, a user can discover many logic and semantic errors that would otherwise have taken many runs with a substantial amount of 'guessing' as to the appropriate correction to make.

Suggestion generation and help:

When a user makes an error, the system attempts to determine the original intent and suggest a correction. The user is given the opportunity to accept or reject this suggestion. A user's statement is parsed and a partial parse tree is constructed. This is analyzed to detect any errors. If an error is detected a nearest match is attempted with a correct partial parse tree segment stored in the system. The closest match resembling a correct structure is used to try to reconstruct the user's partial parse tree segment. This is then traversed to generate the suggestion. The details of this method are fairly technical and are thus beyond the scope of this paper. It is not possible to always generate a suggestion as the omissions from the user's entry makes identification of the intent a guess at best. In this case the system will identify the error to the user as per a typical compiler, requesting the user to correct before continuing.

Static help systems fail to take into account the previous history of the particular users. Users must understand a substantial amount of the problem area to ask for the correct kind of help. A profile could be maintained for each user that would record the usage of both the features of the environment used and of the syntactic constructs mastered in the language. Using this profile, help can be offered in an incremental way. To the complete novice an obtrusive help system is active making suggestions at each fault of the user. As the user accomplishes tasks within the environment, the help system can take more of a back seat and wait to be asked for assistance. This would provide a model closer to the environment existing between a novice user and a human tutor. Of course the ultimate machine based environment would incorporate a complete history of the user and use this to generate intelligent prompting, correcting and encouragement by anticipating the errors and pitfalls of the users. The substantial overhead imposed by such an undertaking does not imply that steps towards this goal should not be taken. A useful environment can be created using a modified history of each user to generate assistance in the form of suggested corrections and examples using the user's incorrect and correct statements. (for a more detailed description of the application of an intelligent help system, see Waugh & Stark, 1987.)

Editing capabilities:

The editor allows a user to enter Pascal programs which can consist of Pascal statements and comments in the form of English text. The editor provides the useful features of automatic indentation in block levels resulting in a pleasing visual presentation of the code. Other features include the ability to cut and paste, use the clipboard to make copies of code for duplication purposes or moving text around and scrolling both up and down the page as well as left and right as the page size is 120 columns wide. Other useful functions include saving of files, resuming a previously saved session and undoing previous commands.

The user interface:

The user interface is menu driven with independent windows, (see fig. 2), for code, program output, variable tracing and communications between the user and the system, (known as dialogue boxes on the Macintosh). The interface was designed to

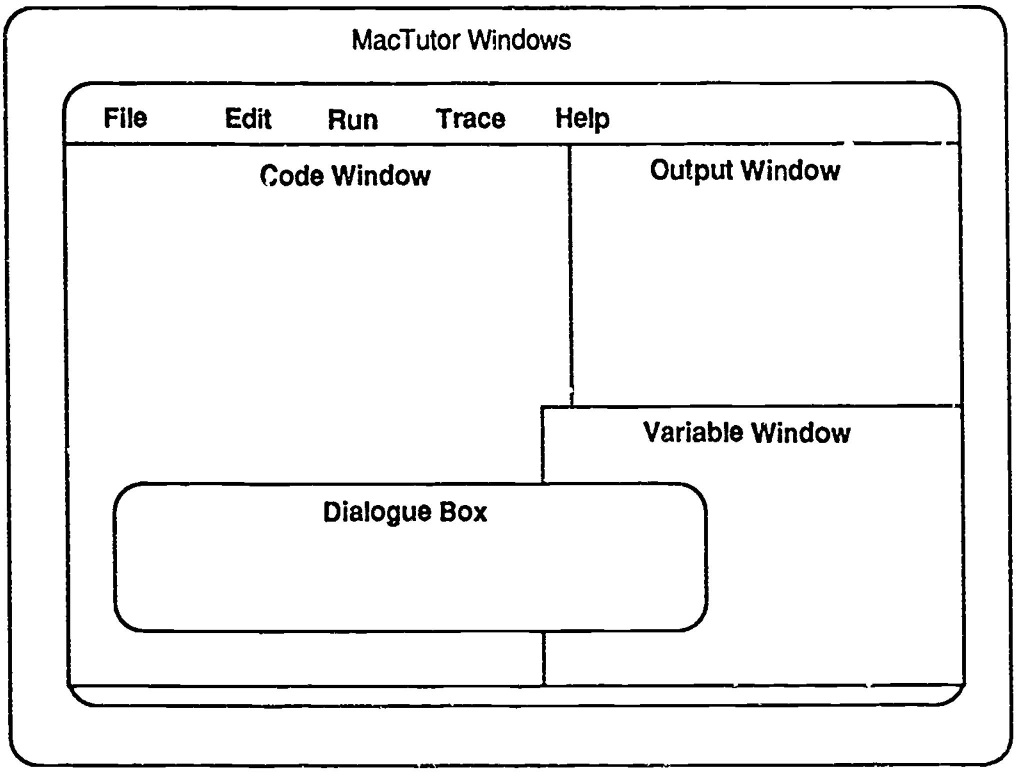


Figure 2

take advantage of as many features of the Macintosh toolbox as possible. This allows for a clean, simple and efficient interface All interactions involve some form of indication using the mouse or typing using the keyboard or menu selection. The amount of typing required by the user is thus minimized. Many users enjoy the novelty and ease of use of a Macintosh, especially when compared to the alternative systems offered by typical operating systems.

Conclusions

In this paper a truly useful Programming Environment for novice programming students has been described. The method of implementation has also been presented with the advantages and lessons learned along the way. The major outcome of the whole undertaking has been the idea to generate a Programming Environment shell that is independent of any particular programming language. Implementations would include a language dependent module being 'plugged' into the shell resulting in a novice Programming Environment for any language that a language module exists. This is a rather ambitious undertaking and is the direction of future research. Other extensions are to include dynamic data flow analysis. Basically, data flow analysis provides a trace of the use and abuse of variables, an example being re-initialising an unreferenced variable, allowing the detection of anomalies in variable usage which can be corrected to provide a more efficient and less error prone program. Dynamic data flow analysis is analysis performed while the program is being created and compiled. This last feature can be accomplished with minimal modifications to the existing environment.

Finally, the environment and its tools can be extended to the professional development arena. This would require the optimization of every facet of the environment as professional software development is not concerned with teaching programming, only the development of software. programming environments, and automated software engineering tools in general, are gaining much enthusiasm as software continues to become larger and more complex.

A closing thought:

'Least is most'
Mies van der Rohe.

(especially when applied to the amount of information a programmer needs to keep in their head and the amount of effort to achieve the end result.)

References

Aho, A. V. & Peterson, T. G. (1972). *A Minimum Distance Error-Correcting Parser For Context-Free Languages*. SIAM Journal of Computing, Vol. 1, No. 4.

Apple Computer Company, (1985). Macintosh Pascal.

Ardis, M. A., Gehling, J., Gill, ' . D., Glushko, R. & Vishniac, E. (1985). An *Assessment of Eight Programming Environments*. Technical Report TR-85-10, Wang Institute of Graduate Studies.

Atkinson, L. V. & McGregor, J. J. (1978). *CONA - A Conversational Algol System*. Software-Practice and Experience, Vol. 8.

Atkinson, L. V., McGregor, J. J. & North, S. D. (1981). *Context sensitive editing as an approach to incremental compilation*. The Computer Journal, Vol. 24, No. 3.

Barstow, D. R., Shrobe, H. E., Sandewall, E. (eds) (1984). *Interactive Programming Environments*. U. S. A., McGraw-Hill.

du Boulay, B. & O'Shea, T. (1980). *Teaching Novices Programming*. DAI Research Paper 132, University of Edinburgh.

Donzeau-Gouge, V., Huet, G., Kahn, G. & Lang, B. (1984). *Programming Environments Based on Structured Editors: The MENTOR Experience*. In Barstow et al, (1984).

Feycock, S. & Lazarus, P. (1976). *Syntax-directed Correction of Syntax Errors*. Software-Practice and Experience, Vol 6.

Ghezzi, C. & Mandrioli, D. (1979). *Incremental Parsing*. ACM Transactions on Programming Languages and Systems, Vol. 1, No. 1.

Goldberg, A. (1983). *The Influence of an Object-Oriented Language on the Programming Environment*. In Barstow et al, (1984).

Graham, S. L. & Rhodes, S. P. (1975). *Practical Syntactic Error Recovery*. Communications of the ACM, Vol. 18, No. 11.

James, E. B. & Partridge, D. P. (1973). *Adaptive Correction of Program Statements*. Communications of the ACM Vol. 16, No. 1.

Kantorowitz, E. & Laor, H. (1986). *Automatic Generation of Useful Syntax Error Messages*. Software-Practice and Experience, Vol. 16(7).

Lesk, M. E. & Schmidt, E. (1975). *Lex - A Lexical Analyzer Generator*. From Unix Programmer's Manual.

Lock, K. (1965). *Structuring Programs For Multiprogram Time-Sharing On-Line Applications*. Proceedings AFIPS FJCC.

Marlin, C. D. (1986). *Language-Specific Editors for Block-Structure Programming Languages*. The Australian Computer Journal, Vol. 18, No. 2.

Moretti, G. S. & Lyons, P. J. (1986). *An Overview of GED, A Language-Independent Syntax-Directed Editor*. The Australian Computer Journal, Vol. 18, No. 2.

Morris, J. M. & Schwartz, M. D. (1981). *The Design of a Language-Directed Editor for Block-Structured Languages*. ACM SIGPLAN Notices 16(6).

Moura, A. V. (1986). *Early error detection in syntax-driven parsers*. IBM Journal of Research and Development, Vol. 30, No. 6.

Parnas, D. L. (1972). *On the Criteria To Be Used in Decomposing Systems into Modules*. Communications of the ACM Vol. 15, 12.

Teitelbaum, T. & Reps, T. (1981). *The Cornell Program Synthesizer: A Syntax-Directed Programming Environment*. In Barstow et al, (1984).

Teitelman, W. (1969). *Toward a programming laboratory*. International Joint Conference on Artificial Intelligence, Washington, May.

Teitelman, W. & Masinter, L. (1981). *The Interlisp Programming Environment*. In Barstow et al, (1984).

Waugh, D. A. & Stark, K. P. (1987). *Applying Intelligent CAL Principles to Help Subsystems for Interactive Software*. Proceedings of the Fifth Annual Conference on Computer-aided Learning in Tertiary Education, University of Sydney, Sydney.

Welsh, J., Rose, G. A. & Lloyd, M. (1986). *An Adaptive Program Editor*. The Australian Computer Journal, Vol 18, No. 2.

David Waugh holds a BSc (Hons) Computer Science, 1985. The University of Western Ontario, London, Ontario. MSc in Computer Science, James Cook University, Townsville. Current Position is a Lecturer, Department of Computer Science, James Cook University. His research interests: Novice Programming Environments, Software Engineering Tools, User Modelling, User Interfaces

Generative CAL and courseware abstraction

Geoffrey I. Webb
Computing and Informatioı Studies
Griffith University

Courseware abstraction is an approach to CAL whereby the lesson author creates a general parameterised CAL lesson that is then applied to many concrete examples. This approach has the following advantages—

- it provides a powerful framework within which to adapt tuition to a student's knowledge and aptitude;
- it encourages the development of detailed treatments of the subject matter;
- it reduces the cost of lesson development as a ratio to student lesson time; and
- it enables large numbers of examples to be made available for individual students.

Generative CAL is an example of courseware abstraction. It is argued that the advantages of generative CAL do not arise directly from the generation of the examples to be examined but rather can be directly attributed to the use of courseware abstraction.

Generative computer-aided learning is a CAL technique that was developed in the late sixties (Uttal, Pasich, Rogers & Hieronymous, 1969; Uhr, 1969). A generative CAL system generates unique problems and presents these to the student for solution. The system must be able to evaluate and critique the student's solution to the problem. This implies that the problems to be generated belong to a class of problems for ᵼ'hich the system has the capacity to generate, evaluate and critique novel instances.

Most generative CAL systems have been in the domain of m⁻˙˙.ematics. This is not a matter of coin ɛnce. Mathematics is one of the few domains for which it is relatively simple for the computer to be provided with the capacity to generate, solve and critique students' solutions to novel problems. Once one leaves the domain of mathematics, relatively difficult AI based techniques are required to create generative CAL. (See, for example, Sleeman and Brown, 1982).

Generative CAL offers several benefits—

- By selecting problems with appropriate attributes it is readily possible to adapt to variations in student's initial competence and learning rate within a domain.

- Because one general treatment of a domain is defined that is then applied to many instances from that domain, the lesson author is encouraged to make the treatment more detailed than if a separate treatment has to be written for each example to be examined as is the case with normal programmed learning.

- Because only the one general treatment has to be written, if the system is able to generate many examples from the domain and consequently the one general treatment is presented to the student a large number of times then the relative cost in authoring time to student terminal time is likely to be very low. This can make the approach very cost effective.

- The student is provided with the opportunity to examine as many problems as may prove desirable, all within the one consistent framework.

However, a careful examination of these benefits will show that none of them result directly from the fact that the computer generates the problems to be examined. Rather, they result from the fact that one general treatment of a domain is applied to many specific instances from that domain. In short, the benefits of generative CAL all arise from the fact that it is a technique that employs an approach to creating CAL that will be referred to as *courseware abstraction.*

Courseware abstraction is a method of creating computer-based courseware that is both time saving for the lesson author and encourages thorough treatment of the subject matter. The author creates once only a detailed abstract CAL treatment of the subject matter. This treatment is parameterised. That is, those aspects of the subject matter that change from example to example are identified, and a variable is provided for each. This parameterised abstract treatment is then applied to a body of specific examples from the subject area. Associated with each example are the correct values for that example for each parameter in the abstract treatment. The result is a detailed treatment of the particular example for the student. The one lesson can then be presented to the student many times, each time using a different example and providing a relevant analysis for that example.

Generative CAL is one approach to CAL that employs courseware abstraction. The vast majority of generative CAL systems have been for mathematics. As stated above, this is not a matter of coincidence.

Generative CAL systems must meet the following requirements.

- They must be able to generate meaningful problems.
- They must be able to solve the problems generated.
- They must be able to monitor, remediate and assist the students' attempts to solve the problems generated.

As mentioned above, although this is reasonably tractable for arithmetic domains, once one leaves the field of mathematics very sophisticated artificial intelligence based systems are required. The development and operating overheads of such systems are prohibitive for most CAL applications.

However, courseware abstraction does not require that the system should generate the problems that the student is to examine. Rather, the course author can create a body of examples that the system can then use, thus avoiding the problem of generating novel problems. This approach has been used with success by a number of systems. It is a key factor in the power of the ECCLES (Richards & Webb, 1985) and DABIS (Webb, 1986) systems. Most AI based CAL systems utilise it. Despite being a key factor in many systems, this approach to CAL has not previously been identified and the role that it plays has not previously been recognised.

As the generative approach has been well documented, the rest of this paper will address non-generative approaches to courseware abstraction. To provide a context for this discussion the next section examines a typical example of an application for courseware abstraction—CAL treatments of the French passé composé. The remaining sections provide a general discussion of the advantages and applicability of the approach and means of implementing it in existing CAL languages.

A computer-based lesson on the French passé composé

The passé composé—the past tense verb agreement system—is an extremely difficult aspect of French grammar. The past tense is marked by the presence in a phrase of one of the auxiliary verbs *être* and *avoir*. Each verb can normally only appear with one of these auxiliary verbs. This aspect of French grammar has many irregularities. That is, for many French verbs it is not obvious which auxiliary verb the main verb takes. This is simply a matter of acquired knowledge.

The agreement rules of the passé composé can be summarised as follows—

- If there is a reflexive pronoun in the phrase as either the direct or indirect subject of the main verb then—
- If the reflexive pronoun is the direct object of the main verb then the verb and direct object must agree in both number and gender. If the reflexive pronoun is the indirect object of a pronominal main verb then the verb must usually be unmarked for both number and gender. This means that it takes the masculine singular form. (The reflexive pronoun is treated here as the direct object of the verb in phrases such as *Elle s'est lavée* but not in phrases such as *Elle s'est lavé les mains.*)
- Irrespective of which auxiliary verb the main verb normally takes, the phrase must contain the auxiliary verb *être.*
- If there is no reflexive pronoun in a phrase and the main verb takes the auxiliary verb *être* then it must always agree with the subject of the phrase in both number and gender.
- If a main verb takes the auxiliary verb *avoir* and there is a direct object in the phrase that precedes the main verb then the main verb must agree with it in both number and gender. If there is no direct object, or if the direct object follows the verb then the main verb must be unmarked for both number and gender.

To further complicate matters, the auxiliary verbs *être* and *avoir* take many different forms depending upon their context, making it difficult for the student to identify whether they are present.

From the complexity of the above description it should come as no surprise that students of French grammar typically experience great difficulty in mastering this aspect of the grammar. It is clearly of benefit to students to provide them with practical experience in identifying whether particular French phrases do conform to these rules and, if not, how exactly they conflict. In short, this aspect of French grammar is a prime target for CAL.

The only practical manner in which to provide this practice is to create a large series of exercises, $E_1, E_2, ... E_n$, each of which examines a separate French phrase, P_i, such that E_i examines P_i.

The standard test and branch approach to the authoring of such a lesson is to create a list of example phrases to be examined and for each of these examples write a CAL segment that provides a detailed exercise examining that example. This will be called the *traditional approach*. This approach can be characterised as the creation of CAL segments $s_1, s_2, ... s_n$ such that s_i generates the interactions with the student required by E_i. As a result, the provision of n exercises will require the creation of n CAL segments.

It would be necessary in such a lesson to examine the following questions, among others, with relation to the examples—

- Is the phrase grammatical?
- What is its main verb?
- What is its auxiliary verb?
- Does the main verb agree with its subject?
- Does the main verb agree with its direct object?
- What is the main verb's number?
- What is the main verb's gender?
- Does the main verb have a direct or indirect object?
- Does the direct object precede or follow the main verb?
- Does the phrase contain a reflexive pronoun?

However, not all of these questions would apply to all of the examples. For example, it would only be important to determine the relative location of the direct object and main verb of a phrase if the main verb took *avoir*.

Each exercise would have to examine only those questions that are relevant to the example on which it is based. Each relevant question would be asked of the student. The student's answers would be evaluated and appropriate feedback provided.

The structure that each exercise might take would be to first enquire of the student whether s/he believes the example to be grammatical. If the student answers correctly then it can be assumed that s/he can correctly analyse the example and another example can be selected and examined. If this assumption is incorrect and s/he has only guessed the correct answer, this would be picked up at a later stage when s/he again has to analyse a similar phrase.

If the student believes that the example is ungrammatical and it is not, each of the reasons why it could be ungrammatical should be examined and the student should be shown how they do not apply to the example. Conversely, if the student believes that the phrase is grammatical and it is ungrammatical, then s/he should be shown exactly how it is ungrammatical. Figure 1 shows a fragment of such an example exercise written in Common Pilot. This fragment tests if the student can identify the direct object in the phrase *Jeanne je l'ai vue hier*. Common PILOT is described in Khieraty & Gerhold (1980).

```
R: Examine whether there is a direct
   object in the phrase.
T: Is there a direct object in
   "Jeanne je l'ai vue hier"?
U: ANSYES
TY: Yes, well done!
TN: Wrong, "L'" is the direct object
   in this phrase.
JN: @P
T:
T: What is the direct object in
   "Jeanne je l'ai vue hier"?
A:
M:%L'%
TY: Very good!
JY: AGREE
M:%JE%
TY: No, "JE" is the subject of this
   phrase.
TY: The direct object is "L'"!
TN: No, the direct object in this
   phrase is "L'"
*AGREE
```

ANSYES is a subroutine that sets the Y conditioner if the student gives an affirmative response and sets the N conditioner for a negative response.

FIGURE 1: A fragment of a standard test and branch lesson on the French passé composé

Let us assume that we are going to provide one hundred exercises for the student. This is far fewer than would be desirable, but it will do to demonstrate the point.

For each exercise there will be a complex schedule of—

- frames of text to be displayed to the student;
- questions to be asked of the student;
- answer evaluation procedures to apply to the student's responses; and
- conditional branching dependent upon the student's responses.

There will be some similarities in this schedule from exercise to exercise, in that the text frames, questions and answer evaluation procedures will be drawn from one pool, but the exact set used will vary greatly from exercise to exercise.

If the traditional approach were used each of these exercises would have to be authored as separate CAL segments. Certainly, there would be some savings in authoring time—due to being able to copy frames, answer evaluation procedures, etc from one exercise to the next—but each exercise would still have to be separately manually constructed by the author.

It is probable that each CAL segment would take at least 100 minutes to author. Assuming that each exercise would occupy approximately 1 minute of the student's time, as each exercise is embodied in a separate CAL segment, this results in an authoring ratio of 100:1. This estimate is probably quite conservative. A ratio of 200 authoring hours to one student hour is not uncommon for this authoring methodology.

On these figures, for 100 exercises the author could expect to spend 10,000 minutes, or approximately 165 hours. All of this for 1 hour and 40 minutes of student lesson time! This amount of time might be feasible for a software house, but it certainly is not for a teacher wanting to provide some remedial exercises for a few students!

This would be a prime target for generative CAL if only it were possible for the computer to generate novel examples and to determine the correct analyses for those examples. The creation of such a system may be currently feasible using state of the art artificial intelligence programming. However, the developmental cost involved would almost certainly far exceed that of the traditional approach.

The alternative approach that is being proposed is to write once only some generalised courseware which can then be used to examine any of a large number of concrete examples from the domain in question. This can be described as the creation of a single CAL segment *S(i)*, which takes as a parameter a description of the item to be examined, and generates the appropriate exercise for the student, E_i. In the case of the French passé composé this means writing a lesson that can examine any combination of verb type and agreement (or lack of agreement) for any phrase. Such a lesson has all the advantages of a generative system with the added bonus that it is feasible to implement!

Figure 2 contains a portion of an example lesson which utilises this approach, again written in Common Pilot. The code in Figure 2 is a subroutine which examines whether the student can identify the direct object in a phrase. This lesson fragment is functionally identical to that in Figure 1. That is, given that the fragment in Figure 2 is being applied to the phrase examined in Figure 1, both respond identically to the student. Despite the fact that the abstracted code can be used in examining any phrase, it is less than twice the size of the traditional code which can only be used to examine one particular phrase. The greater utility of the abstracted code is gained at very little cost.

```
*EXAMDO
R:
R: Examine whether there is a direct
   object in the phrase.
T: Is there a direct object in "$A$
   "?
U: ANSYES
J(D$ = ""): NODO
*DO
TY: Yes, well done!
TN: Wrong, "$D$ " is the direct
   object in this phrase.
EN: NEXTPH
T:
T: What is the direct object in "$A$
   "?
A:
T(%B = D$): Very good!
EC:
T(%B = E$ & E$ <> ""): No, "$E$ " is
   the indirect object.
TC: "$D$ " is the direct object!
EC:
T(%B = C$): No, "$C$ " is the subject
   of this phrase.
TC: The direct object is "$D$ "!
EC:
T: No, the direct object in this
   phrase is "$D$ "
E:
*NODO
TN: Excellent!
TY: Wrong.  There is no direct object
   for this phrase.
EY: NEXTPH
E:
```

ANSYES is a subroutine that sets the Y conditioner if the student gives an affirmative response and sets the N conditioner for a negative response.

C is the subject of the phrase.
D is the direct object of the phrase.
E is the indirect object of the phrase.

FIGURE 2: A fragment of an abstracted lesson on the French passé composé

Clearly the abstracted courseware requires some means of determining the details of the concrete example under consideration. Otherwise it cannot determine what the correct answers are for any particular example phrase. The lesson from which the fragment in Figure 2 is taken reads the phrase and various parameters about it in from a file. Figure 3 shows the parameters that are read in for exercises on the phrases *Jeanne je l'ai vue hier* and *Nous nous sommes arrêtés devant la bibliothèque.*

Phrase	*Jeanne je l'ai*	
	Nous nous sommes arrêtés	
	vue hier devant la bibliothèque	
Grammaticality	GRAMMATICAL	GRAMMATICAL
Subject	*je*	*nous*
Direct Object	*l'*	*nous*
Indirect Object		*la*
bibliothèque		
Reflexive Pronoun		*nous*
Verb	*vu*	*arrêtés*
Auxiliary Verb	*avoir*	*être*
Location of direct object	PRECEDING	PRECEDING
Number	SINGULAR	PLURAL
Gender	FEMININE	MASCULINE

FIGURE 3: Parameters for two examples from the abstracted passé composé lesson.

Using this authoring method, if we assume that it will take the author approximately 40 hours to specify the initial lesson and then approximately 1 minute to specify each example phrase and its parameters, then we have an authoring time of 2500 minutes for the lesson with 100 examples. This is an authoring ratio of 25:1, a fourfold improvement over the traditional approach. It is not really important whether the (probably already excessive) figure of 40 hours for the specification of the abstracted lesson is not accepted. No matter how exorbitantly this figure is inflated, the ratio will still come out in favour of courseware abstraction in the long run. This is because, having written the abstracted lesson and being in the position that each new phrase only takes a minute to specify, the author is unlikely to stop at one hundred examples. The more examples that are specified, the better the authoring ratio will be. This contrasts sharply with the traditional approach where the authoring ratio remains constant no matter how many examples are provided.

Several points should be noted about the lessons of which portions appear in Figures 1 and 2. First, as already mentioned, for the phrases covered by the traditional lesson, both lessons are functionally identical. The traditional lesson only covers three phrases. However, with just these three phrases the author has already written less code (albeit more complex) for the abstracted lesson than for the traditional one. Further, the smaller abstracted lesson can already handle far more than just the three examples that the traditional lesson handles. Indeed, the abstracted lesson can handle any grammatical phrase with a reflexive pronoun as the direct object or with a main verb that takes *avoir*. All that is now needed to make the abstracted lesson complete are short treatments of phrases with reflexive pronouns as indirect objects and of verbs that take *être* and a treatment of ungrammatical sentences. The latter would be approximately the same size again as the existing lesson code. By comparison, the task of the author using the traditional methodology has barely begun with the three phrase example which has been written.

The final point that should be made about the example lessons is that it is not important whether the reader believes that they provide a good or a bad treatment of the subject matter. They are provided only to serve as a concrete example of the difference between abstracted and non-abstracted courseware. Courseware abstraction is not tied to this programmed learning based style of lesson. For example, the GREATERP system involves the application of a general lesson to a sequence of concrete examples. Thus, it is also an abstracted lesson. However, it clearly does not use a programmed learning based approach to CAL. Rather, it utilises extremely sophisticated AI based CAL.

The advantages of courseware abstraction

An improved ratio of authoring to student time is just one of the advantages of courseware abstraction.

Lesson validation is far easier with abstracted courseware. With the traditional approach, the author will never be able to thoroughly determine that no errors lie hidden in some section of code in some exercise or other. With courseware abstraction, any error in the abstracted courseware will quickly come to light, as the same code is used for every exercise. Once the generalised lesson has been debugged, the entire course can easily be validated simply by checking that the parameters passed into the system are correct. If the parameters have been given sensible names then this is straight forward. The author need only scan through a list like that in Figure 3 in which any errors will be transparently obvious.

Another advantage of the methodology is that it encourages the author to provide an exhaustive treatment of the topic. With the traditional approach the author knows that for every aspect that is built into the lesson there is going to have to be separate code written for each example to which it applies. This can result in the author having to choose between providing a thorough treatment of a smaller than desirable set of examples, or a less thorough than desirable treatment of a larger body of examples. This is not true with abstracted courseware authoring because the author only has to provide the one thorough treatment of the domain. This is then applied to all examples examined thus ensuring that every example receives the same level of analysis.

Abstracted courseware is also easier to update. If the author decides that some aspect of the original treatment of the domain under examination should be changed, such a change need only occur in one place in the generalised lesson. Under the traditional approach, any such change would need to be made separately for each exercise to which it applied. This would cause two problems—a) determining which exercises the change applies to; and b) changing the lesson code in each exercise.

Another advantage of courseware abstraction is that it provides a framework in which it is readily possible to adapt instruction to each student's current level of expertise in a domain. If the CAL system is able to identify relevant features of the examples that it is examining, such as their relative difficulty, and is able to judge from the student's performance how well they perform in relation to these features, then it is relatively straight forward to select examples for examination that are appropriate for the student's current needs. For instance, if the examples are graded with regard to difficulty, and the system can determine that the student makes no errors when examining examples at difficulty level one, makes errors for 50% of the examples at difficulty level two and 90% of the examples at difficulty level three then it will be able to determine that it should concentrate on examples at difficulty level two. All that it need do is start at the easiest difficulty level and work its way down to the more difficult levels, remaining at each difficulty level until the student achieves a set success rate.

But difficulty level is not the only or the most powerful distinguishing feature that could be taken into account. Consider the abstracted passé composé lesson. If the student is not making errors for examples that take the auxiliary verb *avoir* but is for examples that take the auxiliary verb *être*, then it will be quite trivial for the system to identify this fact and to concentrate on the examples for which the student needs further tuition—those which take the auxiliary verb *être*. All of the information that it needs in order to determine this—that is, the auxiliary verb for each example—is already explicitly stated and thus readily accessible.

A final advantage of abstracted courseware is that it greatly facilitates the analysis of a student's performance during the lesson. If a simple record is maintained of the questions at which the student makes mistakes, then it will be readily possible to determine what difficulties the student is experiencing. For instance, if the student often specifies the wrong word as being the main verb in the phrase then it is clear that they are having difficulty identifying the main verb. The standard test and branch approach makes this information far more difficult to obtain as there is no clearly defined structure by which to assign identity to similar questions in different exercises. At best, this becomes another chore for the author to look after. At worst, this information is simply not collected.

The applicability of courseware abstraction

Courseware abstraction is clearly not a methodology that can advantageously be applied to all domains for which a computer-based lesson may be written. Rather, courseware abstraction only comes into its own when large numbers of concrete examples are to be examined in terms of a general theoretical framework.

An obvious area in which this applies is grammar. Typically languages have very complex grammatical systems for which it is advantageous for the student to examine large numbers of specific examples within a general framework. The passé composé lesson is an example of just one such lesson.

However, languages are far from the only domains to which such lessons may be applied. A lesson has been developed under the DABIS system (Webb, 1986) that examines the leading theories in the mind/body debate within philosophy. The lesson identifies a number of central questions within this debate and examines how each key position in the debate has a different set of answers to these questions.

Another possible use for the methodology is in the control of practical classes where the students must follow a strict order of enquiry to reach some conclusion. For instance, the students could be given a chemical solution and then the program could guide them through the step by step process of identifying its chemical composition. The same lesson could manage the examination of any of a large number of chemical solutions.

Courseware abstraction is a methodology of wide but not unlimited application.

Courseware abstraction in existing CAL systems

As is demonstrated with the Common Pilot lesson from which the fragment in Figure 2 is taken, courseware abstraction is possible using at least some existing CAL languages.

All that is required for a CAL authoring system to allow courseware abstraction is for it to support some method of specifying variables and then assigning them different values for different examples; and for it to allow conditional branching dependent upon the value of those variables.

Probably the simplest way to implement this in most existing CAL languages would be the approach taken in the lesson fragment in Figure 2. That is, to define a set of variables which are then read in from a file. For each example that is presented a new set of values is read into the variables from the file. Once a lesson has been written, a text editor can then be used to create and update the input file.

However, for courseware abstraction to be possible, it is not necessary for a CAL language to support files. If the language supports variables then it is possible simply to assign values to them within the program. Thus, each time that an exercise begins a new set of values are assigned to the variables.

Although courseware abstraction is possible using many existing CAL languages, specialised systems, such as ECCLES (Richards & Webb, 1985) and DABIS (Webb, 1986), that take advantage of the special features of the methodology increase its benefits to the author.

Conclusion

Courseware abstraction provides a methodology for the authoring of computer-based courseware which can create detailed lessons of a type not feasible through other approaches due to their huge cost in authoring time and difficulties of maintenance.

The benefits that courseware abstraction have to offer have already been demonstrated by the success of generative CAL. Among the attractive features of courseware abstraction are reduced authoring time; the provision of a convenient framework for adapting tuition to the student's level of expertise; improved lesson verification; improved lesson modifiability; and improved student analysis.

Although courseware abstraction has been used in many previous CAL systems, there does not appear to have been any appreciation of the fact that it was courseware abstraction to which many of the desirable features of these systems could be attributed. This is particularly notable in the case of generative CAL where the benefits of the approach have been attributed to the generation of the problems to be examined whereas they can actually be attributed to the use of courseware abstraction.

It is to be hoped that a better understanding of factors underlying the success of these approaches will in turn lead to the development of yet better approaches.

Acknowledgements

The research presented in this paper grew out of the ECCLES project which was founded and led by Tom Richards. Thus, the ideas presented herein have directly evolved from Tom's original inspiration for ECCLES. I am also deeply indebted to Bob Hooke and Roly Sussex who advised me on the finer points of French grammar during the preparation of this paper. The lesson fragments in Figures 1 and 2 are based on a lesson written for the ECCLES system by Bob Hooke.

References

Khieriaty, L. & Gerhold, G. (1980). *COMMON PILOT Language Reference Manual.* Bellington W?: Western Washington University.

Richards, T. J. & Webb, G. I. (1985). ECCLES: an "expert system" for CAL. In *Proceedings of the 1985 Western Educational Computing Conference,* Oakland, C.A., pp. 151-157.

Sleeman, D. & Brown, J. S. (Eds). (1982). *Intelligent Tutoring Systems.* London: Academic Press.

Uhr, L. (1969). Teaching machine programs that generate problems as a function of interaction with students. In *Proceedings of the 24th National ACM Conference,* pp. 125–134.

Uttal, W. R., Pasich, T., Rogers, M. & Hieronymus, R. (1969). *Generative Computer Assisted Instruction.* Communication 243, Mental Health Research Institute, University of Michigan.

Webb, G. I. (1986). *Knowledge Representation in Computer-Aided Learning: The Theory and Practice of Knowledge-Based Student Evaluation and Flow of Control.* PhD thesis, La Trobe University School of Mathematical and Information Sciences.

2 Text-dominated instructional material usually defines and prescribes the learning experience and seldom draws on the experience or potential of the learner.

3 There is an assumption that self-motivation of the remote learner will help him or her overcome the lack of learning resources.

4 There is not enough understanding or appreciation that a remote learner may not have participated in a course for the sake of merely getting a qualification.

A videotex-based AESNET

King and Foste.'s chapter is in a book on study centres (Castro, Livingston and Northcott 1985) published in August 1985, the year that saw the launching of Australia's videotex system (Viatel), first satellite (Aussat), two public electronic messaging systems (Minerva and Telememo, now merged to form Keylink), and the year that the International Council of Distance Education held its Thirteenth World Conference at Melbourne. The Hon. Susan Ryan, then Minister of Education, in her opening address to the 600 delegates from around the world, announced the Government's "very important" commitment to Edutel, an educational network on Viatel. The vague expectations of the Government and the external studies fraternity were that Edutel would develop into a national network like Prestel Education in the United Kingdom, and that it would have a comprehensive national course directory which could be continuously updated by all participating institut.ons. In other words, there would be an Australian external studies network (abbreviated in this paper as AESNET) built on a publicly available and simple to use videotex system. Study centre personnel, by signing on to Viatel, would be able to show students the courses on offer, and perhaps even offer online counselling.

Such expectations were never realised and it is unlikely that they will be. Videotex is a simple technology with a pleasing mixture of colour, graphics and text. Its menu approach, numerical branching technigue, and split rate access (sending at 75 bps and receiving at 1200 bps) make it slow and cumbersome when compared with command-driven systems. It is therefore not effective or efficient for large databases, mail and sophisticated information retrieval; but is a promotional medium more suitable for business or commercial applications.

Most institutions and home-learners who rushed to subscribe to Viatel soon discovered its educational limitation and as a result, any in[illegible]l euphoria and enthusiasm towards a national network based on Viatel have vanished. There are also different environmental factors in Australia and the United Kingdom to account for the failure of educational videotex in the former and the relative success in the latter (Castro 1986). So one could say that the notion of a videotex-based AESNET was virtually shelved earlier this year when the Government finally withdrew its limited support in early 1987 to the Australian Caption Centre, a non profit-making organisation appointed to develop Edutel. Edutel has since been sold to a commercial videotex bureau. Recent reports, however, indicate that the notion is back in vogue with the Commonwealth Tertiary Education Commission and some selected Technical and Further Education sites will be asked to develop and manage such a network for use by all external students and institutions.

It would be useful, therefore, to pause at this point and ask what can such a network hope to achieve. Is it to replicate the current conventional support services which, in the view of King and Foster, are more directed to meeting just the needs of the institution? Or should it try to provide a caring and more efficient environment which some would regard as beneficial for individual learning?

These are difficult questions which do not seem to have been widely or seriously contemplated or addressed by people who have expert knowledge about the strengths and weaknesses of educational technologies and, more importantly, an appreciation of the crucial issues of the need for technology management and user education. While I certainly do not pretend to have all the answers and can only speak from limited, personal experience as an instructional designer, an on-line network manager, and a researcher of information technology, I do not subscribe to the view:

- That an AESNET has to be based entirely on computer technology or even on a particular kind of system; and
- That a nationally directed and managed network will attract a workable user base from within and outside the tertiary institutions.

The philosophy of a home-based network for tertiary distance and open education

A different philosophy, and in my view a more realistic approach, would have been to visualize the student support system as beginning at home - the home of the student and the home of the institution. If institutional decision is that a technology-based support system should complement the current conventional one, then I would propose a serious consideration of the following four tasks:

1 That there should be a thorough review of current computer resources within each institution to see if they could be integrated to provide a reasonable electronic infra-structure of student support.

2 That this structure be managed by a small team of people who are technically competent, can take on the role of a mentor and a catalyst, and who are personally committed about their facilitative and gatekeeping responsibilities towards the student users of such a network.

3 That approaches be made to established and operational national, state and commercial networks (e.g. ACSNET, VICNET, CSIRONET and KEYLINK) to request for interconnection to institutionally-based networks to form a national AESNET at some stage in the future.

4 That an institution start training both staff and students to help them become effective users of the network.

All four tasks have to proceed concurrently but criss-crossing in a timeline that may stretch to three, four or even five years.

For the first task, I shall now quote some examples of using computer and telecommunications technologies at Deakin University. The purpose is not to publicise how advanced Deakin University is but to draw attention to then fact that they are the sort of resources or embryonic services already existing in many tertiary institutions and which in time will be used by an increasing number of staff and students at a rate commensurate with how they are introduced or promoted.

The following is a summary of the computing resources at Deakin; machines and connections which are irrelevant to the current discussion have been omitted for the sake of clarity.

1 Machines:

- Charlie (Gould PN6080, UNIX central machine for academics and students)
- Alice (Data General Eclipse S/160 for Library)
- Microcomputers (app. 300 units in microlabs and staff offices, mainly IBM-

compatibles for Schools of Sciences. Management, Architecture and Library; and Macintoshes for Schools of Education, Social Sciences, and Humanities)

- Terminals (app. 300 units)

2 Connections

- The bigger machines and terminals are linked by cables to Micom and Ethernet ports.
- Microcomputers are nearly always used as stand-alones and currently few of them have modems or communications software or are in a local area network. Only the Library microlab have Appletalk (for Macintoshes) and EasyLAN (for Olivettis, IBM-compatibles) for linking to printers.
- Staff and students can dial in from their offices or homes Charlie using STD and Austpac but permission must be previously granted and payment guaranteed. The Austpac PAD has eight lines, one for dialling out, the rest mainly for dialling in by external students on the Graduate Diploma of Computing course.

Staff members and students must register on Charlie first before being give a disk or directory quota and permission to use the centrally installed programs and access to certain database, mail, news and printing facilities. Once that is done, I can do any of the following things by using the terminal in my office or the microcomputer in my office or at home with a modem, a communications program and a telephone.

1 Exchange mail with my colleagues and students.

2 Exchange mail with people in Australian institutions (free of charge) with people in overseas computer networks (at reasonable rate) via ACSNET, of which Deakin is a member.

3 Read and contribute to Deakin news.

4 Read and contribute to Australian news (free of charge) and overseas ones (at reasonable rate) via ACSNET.

5 Access the library catalogue and search for books, then phone to arrange for a loan, or send electronic mail to a librarian to ask for books to be posted to me to save time and a trip to the Library.

6 Send a request to a reference librarian to ask him/her to do a database search for me and forward the results or abstracts of relevant titles to my address on Charlie, which I then process and refile in to a personal database.

7 Run statistical programs or CAL programs without having to buy them pirate them at the risk of breaking copyright laws.

8 Have a word-processed file printed on one of the expensive laser printers attached to the big machines.

9 Build a bibliographical database with search software and invite some colleagues or students to use it or to add to it.

10 Use the Austpac line to dial out to external systems on which I am a permitted user (e.g. Keylink, ERIC, etc.)

11 Ask my students what they think of a course through personalised mail or a questionnaire.

12 Conduct computer conferences to simulate tutorials, either in real time or in a store-and-forward messaging mode.

13 Get students to produce an on-line journal to exchange their writings and thoughts on their course.

All these things have not yet been perceived in Deakin as a kind of package services which we can offer to our external students for every course or program, but one could see the potentials for home access by a remote student if arrangement can be made to allow him or her to dial in using a modem linked to a microcomputer. Now if one compares these services with the kind which has been suggested should or could be mounted on a videotex AESNET, one could immediately see the advantages of having a home-grown and home-based network which is capable of being customised to suit the personal needs of the individual student who does not want to travel to a local study centre (usually after work, at night or during weekends) which can be 6 or 60 kilometres away. Direct and almost instantaneous communications can take place between an instituti , a teacher and a student without the constraints of time, place and distance.

The Deakin scenario reflects the potential offerings of a well-endowed institution, which may even be able to bear the telecommunications costs of the students. For institutions which prefer a low-cost front-end setup, more modest designs may be used. For example, Murdoch University has a small bulletin-board type network built on an IBM-compatible microcomputer (Atkinson, 1986, 1987). Brisbane College of Advanced Education, on the other hand, favours a different approach altogether by registering their students and staff on a network built on Keylink. North American and European distance-teaching institutions have gone a slightly different route again by purchasing off-the-shelf computer conferencing systems because they are considered to be easier to handle by staff and students who may not be too computer-sophisticated. (Kaye 1985, 1987; Harasim 1986)

The role of an external studies network

These big nd small networks all share the following c jectives and features:

1 Institutionally-based and home-accessible.

2 A more direct communication channel between the student and the teacher which is not constrained by time, distance and place.

3 Promoting peer group interation to remove the feeling of isolation.

4 Making available resources and opportunities which would have been beyond the reach of the home-learner.

It is a little early to know it they will help remove all or most of the disjunctions as observed by King and Foster, but they are commendable first steps to take to cultivate a caring environment. And if members are properly led and encouraged, they will find imaginative and innovative ways of using even the most modestly-resourced networks, as it has been found in some research projects (Castro 1987).

There is now a small but increasingly articulated view that computer-mediated communication systems should be adopted in external studies. Bates (1986, p42) explains why he sees it as more valuable than the promised wonders of computer-assisted learning:

The structuring of the teaching is not contained in or restricted by the architecture of the computer, but developed and negotiated between learners and teachers.

To this, it could be added that learning is enriched and multiplied by the sharing of wisdom and experience among the students themselves which is far more important than making the institutional grade (Castro 1987). The facilitating of good teaching and learning must remain the single most important role of any external studies network.

The "messages" of ACSNET

I feel that it would be discourteous of me, delivering a paper with a title like AESNET, not to pay a tribute to the great network mentioned in passing earlier which started here in the University of Sydney.

ACSNET (Australian Computer Science Network), cited by Quarterman and Hoskins (1986) in their classic paper as a "cooperative network", is based on the Sydney UNIX Network (SUN) software primarily developed by Piers Dick-Lauder and Robert Kummerfield of the Basser Department of Computing Science. The first two machines connected were from the Universities of Sydney and New South Wales in 1979. ACSNET currently spans the continent with approximately 350 hosts at universities, institutes of technology, CSIRO divisions, Telecom Research laboratories, and a number of private and developmental organisations; and operates over fixed lines, leased lines, Ethernet, CSIRONET and Austpac.

The purpose of ACSNET is to support mail traffic, file transfer, print jobs, or anything that can be transferred in a string of bytes. True ACSNET hosts are UNIX machines, but there are gateway programs to make it also possible to communicate with non UNIX systems. There are several UUCP (UNIX to UNIX COPY) gateways to North America and Europe from Melbourne. The underlying transport protocols are also used to support the USENET news transmitted from America to Australia via the Midas X.25 service each evening and paid for by the Division of Information Technology at the CSIRO (about $22,000 a year). The SUN software (now being upgraded to include X.400 capability) is distributed for a nominal charge to academic and research institutions. There is no central administration, each host is usually managed by a small team of experts who also educate or help the users if difficulties are encountered. The original developers and Robert Elz from Melbourne University act as national co-ordinators. There is no government funding; each host pays for its own link.

The addressing format of ACSNET is simple to remember and use: username@host-name.oz. Oz stands for ACSNET, which could be regarded as a subdomain of AU, the domain name for Australia. Messages are passed along on ACSNET, from one host to another, possibly utilising intermediate hosts, in a store-and-forward manner. Messages sent within Australia is free, overseas messages are charged at the rate of between two and three cents per 64 bytes.

The history and the operational mode of the ACSNET have some important lessons for our discussion of AESNET. It shows that a network stands a much better chance of success it it starts small, is designed and managed by people who are competent and committed about a set of clearly defined and workable objectives, and who take the effort to persuade others to join them to create a critical mass of users. The ACSNET's decentralised model also means that there is no complex, overbearing top management to stifle local initiatives and experimentation, and that the naive and new user will have the additional advantage of having to learn only the host system and be able to correspond with others on remote systems without hassle, thus reducing the learning curves and operating costs but gaining in self-confidence and efficiency.

References

Atkinson,R.J. (1986) Computer bulletin boards for distance education students. Proceedings Ed Tech 86, International Educational Technology Conference and Exhibition, Perth, Western Australia, December 1986.

Atkinson,R.J. (1987) Development priorities for computer communications in distance education. Paper for ASPESA 8th Biennial Forum, University of New England, Armidale, NSW, 27-31 July 1987.

Bates,T. (1986) Computer assisted learning or communications: which way for information technology in distance education? *Journal of Distance Education*, *1*(1), 41-58.

Castro,A.S.(1986) Videotex in tertiary education: the missing links. A keynote address delivered on 1 July 1986 at the National Forum during the 2nd Asian-Pacific Videotex Conference and Exhibition, Sydney.

Castro,A.S. (1987) Teaching and learning as communication: the potentials and current applications of computer-mediated communication systems for higher distance education. Paper prepared for the UNESCO Higher Distance Education Meeting, 6-11 September 1987, at Deakin University, Geelong, Victoria 3217, Australia.

King,B. & Foster,A. (1985) A South Australian perspective on student support. In A.S.Castro, K.T.Livingston & P.H.Northcott (eds.) *An Australian casebook of study centres in distance education*. Victoria, Distance Education Unit, Deakin University, p99-112.

Harasim,L. (1986) Computer learning networks: educational applications for computer conferencing. *Journal of Distance education*, *1* (1), 59-70.

Kaye,T. (1987) Introducing computer-mediated communication into a distance education system. To be published in the *Canadian Journal of Educational Communications*, 16(2).

Kaye,T. (1985) *Computer-mediated communications systems for distance education*. Report of a study visit to North America, September/October 1985. Project Report CCET/2. Milton Keynes, Institute of educational Technology, The Open University.

Quarterman,J.S. & Hoskins,J.C. (1986) Naotable computer networks. *Communications of the ACM*, *29*(10), 932-971.

An automated library and borrowing system (ALABS): Implications for CAL

R . J. Clarke and L. Athanasiadis
Microcomputer Laboratories
University of Wollongong

ALABS was designed as an operational control mechanism for the loan of floppy-based software, hardware, and audio-visual materials to students and staff, and the reservation of microcomputer resources in a spatially distributed microcomputer centre. This paper describes the experience of the Microcomputer Laboratories University of Wollongong, which consists of 5 laboratories of 20 microcomputers each, plus another 20 scattered over the campus. ALABS is designed to provide software utilization data by student (for example CAL courseware), laboratory loading statistics, activity rates, as well as maintaining an accurate inventory of audio-visual, software and hardware resources. ALABS is suitable for computing facilities where the diversity of machines and applications means that a single network structure is neither desirable nor feasible. The impact of ALABS on the provision and management of large scale CAL facilities is discussed.

Two sets of problems beset individuals and organizations attempting to provide CAL facilities in the tertiary education- environment. The first set of problems is the determination of the type and variety of CAL products to be supported (software, hardware, audio-visual aids and printed materials) by academic and computer support staff. Attendant problems include accessing the need for new software, determining the sources and alternatives available for software acquisition, making the choice between the internal development or purchase of software, installation and implementation issues, evaluation of the software, the rating of candidate systems and the ensuing costs and contractual obligations of software acquisition and maintenance (Borovits 1984, 66-9).

The second major set of problems addressed in this paper is that of appropriate mass distribution (delivery) and management of CAL and other resources after they have been acquired or created. Potential problems include the structuring of laboratory time, managing loads through laboratories, determining if items on loan are overdue or held by staff, finding where a particular person is in the centre, managing machine bookings, and the allocation of resources to these activities. In order to solve them in the general case the Microcomputer Laboratories restructured its methods of material's distribution and management, producing a computer assisted solution called ALABS (Automated Library And Borrowing System).

Facility Description and Physical Layout

The Microcomputer Laboratories provides teaching support for all six faculties at the University of Wollongong, including Commerce, Education, Science, Engineering, Humanities and Arts. Subjects supported range from machine language programming to creative arts. The facility undertakes consultancies (internal and external to the University), provides computing expertise for academic and research activities and runs courses for university staff and the general community.

The Microcomputer Laboratories consists of 5 teaching laboratories with 20 microcomputers each, an Operations Room, a Data Preparation Room, Workshop and a Systems Resources Room. Three of the laboratories contain Sperry PC Model 25s, with 512-640K RAM, dual 5.25" floppy disk drives and Colour Graphics Adaptors. Two of the Sperry laboratories are configured using 2 Sperry USERNET 286 LANs each with a Sperry PC IT as a file server with 30 Mbytes and 44 Mbytes of secondary storage apiece. The remaining two laboratories consist of Apple 2e machines with 128 Kbytes RAM, dual 5.25" floppy disk drives. One of these laboratories has colour displays and the other monochrome.

The laboratories are spatially distributed (separated by large distances). There are no direct lines of sight into any of the laboratories from the Operations Room. Six staff members are employed, only a maximum of three are available to periodically check the laboratories.

CAL Resource Management in a Tertiary Environment

There are three vexing and interrelated issues in CAL Resource Management within a tertiary education environment. They are the nature of the CAL resource itself, the organizational history and activity profile of the facility, and the lack of general support tools (network and stand-alone) for CAL resource management.

CAL resources have the following attributes

CAL resources are often multi-media in nature. Video disks and tape, kits including workbooks, slides, and manuals as well as the more conventional software based tutorials, drills and simulation can constitute CAL resources.

CAL resources are expensive. In an attempt [illegible] duce the long term expense of commercially available CAL, computing facilities may opt for internal production. The tools required to internally produce CAL products (bulk software, high grade documentation etc.) are also expensive. In the tertiary environment demand for CAL resources always outstrips the available funding.

Pressures exist to maximize usage of the CAL resource yet minimize damage and loss. These requirements are in conflict.

Organizational history and activity profile

Computing facilities in the tertiary environment evolve and can change function and scale in unexpected ways. For instance, the facility from which the Microcomputer Laboratories evolved served as the administrative data processing facility for the then College of Advanced Education. Facilities consisted of a Perkin-Elmer 7/32 minicomputer and nine Apple II+ units (used for teacher training and to offload word processing users from the minicomputer). Upon federation of the College and the University, the facility possessed 60 Apple II+ and IIe microcomputers and had divested itself of the 7/32 minicomputer. An emphasis on computer awareness replaced the administrative role, and the internal support function was replaced by student training.

Equipment funding is often sporadic and linked to factors outside the facilities' control (for example, resource statements associated with new course proposals). Forward planning becomes difficult.

In the case of microcomputer systems the market is so volatile that both hardware and software may become obsolete (this is a consideration in Information Systems area). The systems may not be supported by the vendor, or in the worst case, the vendor may have ceased trading.

The diversity of activities and course requirements conflict with CAL Resource Management priorities. For example, Education users of our facility would like to train teachers on as many microcomputer systems (those on Government contract) as are likely to be encountered in schools. Business Systems users would like to keep up with the corporate state-of-the-art. Users in general want to increase system diversity. Because centre management are charged with the support and maintenance of existing CAL resources, if not the production of them, time and effort is reduced if systems are as homogeneous as possible.

Lack of CAL resource management tools

Networking existing machines and relying on the network management system IS NOT a solution to CAL resource management.

Network hardware and softwa.e are very expensive, especially with the number of workstations required in the tertiary environment. Coupled with the diversity of hardware generally found in tertiary computing facilities, this means that it is unlikely that all machines will be networked.

Network management tools assist with network software and global resources only. The network tools required in a tertiary environment are different from business priorities.

Business networking systems are created for a small and static number of users. Network management tools reflect this. For example, USERNET 286 network contains a function to allow users to be added to the system, but only one at a time!

There are apparently no commercially available stand-alone microcomputer based systems suitable for CAL management tasks. The closest commercial analogues follow methods proposed in, for instance, the Operations literature (Fried 1979, 222-260), including Information Centres (Perry 1987, 127-147), Service Centre Management in general, and the PC Resource Centres in particular (Jackson 1985, 151-154). One package recently reviewed (Syk , 1987), is called Micro Resource Management (Atrium Information Group). Again, business resource management priorities are not those of an educational CAL provider. This system places emphasis on keeping track of machine location, current configuration, and financial management of microcomputers within a business (Sykes, 1987).

Whereas the business microcomputer sector has arguably been of greatest benefit in the production of relatively inexpensive and fully functioned software applicable as educational tools (Sperry 1985a, 1985b, 1985c), the means of managing and distributing these resources in a tertiary education environment are not available off the shelf. This necessitates the creation of in-house tools tailored for this purpose. With dramatic increases of student numbers and transactions (see Table 1), new clients resulting from the federation of the Institute and the University, and the increasing timetabling demands of existing departments, it was necessary to provide new solutions to cope with the increased throughput. This necessitated the restructuring of laboratory time, borrowing privileges, and development of a computer system (ALABS).

Restructuring of laboratory time

The restructuring of the timetable consisted of dividing time into three categories. Free time or 'slack', is time not directly utilised for university courses. It is used for spreading abnormal student loadings, unscheduled classes, external short courses, and hardware maintenance. Free time is not allocated to a given subject.

The regular usage of laboratory time by students is divided into two forms. Supervised laboratory time occurs when a tutor or lecturer has face-to-face contact with students in a class room role. The second form of borrowing is by a student for individual study purposes. This is called unsupervised laboratory time. Time slots are assigned to each class generally in a ratio of one hour unsupervised to every two hours supervised. Unsupervised lab times are reserved for a given course. If over the period of several weeks these machines remain idle, students not in the course are allowed to use machines. If this is the situation then lab staff allocates the time as equally as possible. Special notice is given to the distribution of unsupervised evening access times (5:30pm - 9:30pm). Care is taken to ensure that evening students have equal access to the facilities.

Allocation of the time resource in a spatially distributed microcomputer centre is analogous to other timetabling and optimum packing problems (finding an optimum allocation for limited resources). Not only can there be a feasible solution, but a best arrangement. A collected slack method, where Free Time is gathered together (Brown 1972, 3), appears to be applicable to this centre. Although the timetable is currently prepared manually prior to the start of each session, the availability of this method bodes well for the future automation of this task.

Unique identification of items and users: Barcoding

Similarities between the control measures employed in libraries and supermarkets led to the adoption of the machine readable barcode as the method of uniquely identifying all items. Barcoding is an ideal mechanism for keeping track of numerous items required of ALABS.

The Medium Density Code 39 Barcode Standard is used on all types of items in the centre (software, hardware, and audio-visual). This barcode format has a line of text under the code itself. The coding used enables the item to be identified directly from this text. The three types of barcodes are delineated from each other by the value of the first character. A Sperry PC and an Epson RX-80 printer are used with the Barcodes on Demand package (Barcode Technologies) to print item barcodes onto adhesive address labels.

Software Barcodes are affixed to the diskette and covered with contact to prevent removal and deterioration. Surprisingly, the use of a contact wand does not appear to lead to mechanical damage of the disk media. A back-up barcode is kept in a software barcode folder. The folder is useful for quick entering of a class loan. Audio-visual items (books, journals, videotapes and slides) and hardware items have barcodes attached to them in an appropriate format.

Machine resources are barcoded as well. In order for a machine to be used, a student must present their student card to the Operations Room and receive a lab pass along with any software they may need. Each laboratory has a set of colour coded cards, one per machine. The machines are numbered with barcodes. This allows clients to be assigned to a particular machine for the duration of their visit and ensures against multiple bookings of machines. A machine is available only when its card is held by the lab staff and when that laboratory does not have a tutorial class currently assigned to it.

A High Density version of Code 39 Standard is used by University Administration on the student identification cards. The barcode light wand used for ALABS (MR 300), can read both high and medium density barcode formats without a configuration change. The barcode on the student identification card is used to record transactions for the student, and the 'labpass' provides their location. Medium Density

Code 39 barcodes are generated as needed for staff and tutors who use the centre.

ALABS System

ALABS was targeted for an Apple IIe system, running dBase II under the CP/M operating system, as most of the required hardware was already available at the Laboratories. The initial targeting of an Apple IIe gave the advantage of a lower entry cost (see Diagram 1 for ALABS hardware). Mindful of an eventual target to another machine, the barcode wand purchased was a stand-alone unit.

ALABS was designed as a system to control and monitor the borrowing of lendable items (software, audio-visual items, and hardware) for the Microcomputer Laboratories in the absence of both a single network structure and network usage control facilities. ALABS currently consists of 178 modules totalling approximately 582 Kbytes of dBase II, with some Z-80 and 6502 machine code routines. The system utilizes 38 permanent databases and several temporary databases.

Two main modes are provided in ALABS. The first mode is an operational mode; the second is a supervisor mode. Super-visor mode groups together all powerful or sessional management functions. These include erasing the student and tutor databases (occurs at the end of a session), deleting staff members, deletion of the student sessional borrowing file, viewing deleted library records or archived repair records, checking and re-indexing databases, batch printing of student sessional borrowings, printing staff and tutor databases, initialising and altering laboratory bookings or adding/deleting machine bookings.

In the operational mode, loans and returns may be made (see Usage Scenarios section for more detailed descriptions). Interactive functions include determining items currently on loan by students, staff, tutors and long term staff loans. Included is a count of the active student loans. Students and tutors may have extra items appended to current loans.

The system regularly and automatically determines if any items on loan are late. Overdue student and staff loans may be checked. Offending students who do not return items within the required period are automatically placed on an offence database. Depending on circumstances their account with the centre may be frozen for a given period, so that they cannot use the facilities outside of supervised time slots.

Machines in laboratories may be booked in advance according to loading demands (due assignments!). Weekly timetables for all laboratories can be initialised. Functions are available to assign a laboratory time slot as free, supervised, or unsupervised. Given other constraints, individuals may book machines no later than a day in advance. Terminal booking sheets can be printed. Booked machines are released if the student has not arrived within the first 15 minutes of their time slot. In operation this section is reminiscent of a spreadsheet.

An Auxiliary Commands section allows some of the reporting functions of the supervisor mode to be performed in the operational mode. A hardware maintenance and repair system and a function to change a computer's status (operation or under repair) is provided. Also provided is a Batch Command Processor which allows long reports to be printed overnight.

Complete library functions are provided for item databases. These can be quickly edited to reflect changes in stock. Software records consist of a barcode, name, version, publisher, commissioning date, license number, machine usage and disk media variety. Audio-visual records are defined according to type of item. Journal articles and books are recorded with author, date etc. in the manner of a library. If the item is a videotape, then the record consists of the title, presenter, duration, and category. If the item is a set of slides, then the number of slides are also recorded. Hardware records include barcode, item, model, supplier, serial number, machine number, commissioning date, location, cumulative downtime, and the last date down.

Extensive database back-up facilities are provided. These include all Library databases (Software, Hardware, Audio-Visual), Deleted Library database, Staff, Students, Tutors, Loans (Staff Loans, Staff Permanent Loans, Overdue Student Loans), Sessional Student Borrowings, Offence, Daily Machine Software Usage, Hourly Loans Counts, Hourly Borrowing Rate, Borrowing Duration, Down Time, Down Equipment, Repaired Equipment, Faculty Usage, and New Students databases.

The system may be temporarily closed down for forms printing or maintenance. A complete close-down is issued at the end of daily transactions. After back-up of transactions onto floppy disk, ALABS can be left unattended. At this time daily reports and housekeeping functions are performed. Functions performed include the printing of a transaction file, updating sessional borrowing file, appending New Students to the student database, the calculation of hourly loan rates, faculty usage figures, the printing of machine usage figures, and processing overdue student and staff loans. Also, expired offences are cancelled, and a report on which files to back-up before starting operations the next day is produced. The machine will power itself down after close-down procedures have been performed.

Most of the data entry for ALABS consists of single keystroke menu navigation followed by three or more barcode reads (student card, 'labpass' and at least one item). A keyboard entry mode is also provided in the case of any adventitious failure of the barcode wand.

Usage Scenarios

Unsupervised Student Borrowing

When borrowing an item the student number from the student card is read using the barcode wand. If the student is not on the sessional database, the operator is alerted and the student manually provides full details on a Sessional Student Borrowing Form. The details are entered into ALABS by the operator, and loans can be made. The student card is retained by centre staff member for the duration of the loan. A machine/lab pass is read and given to the student. Requested items are found and each barcode is read. Machine allocation and loan duration is currently two consecutive hours. Students may re-borrow at the end of each two hourly interval if demand for the item is low. Borrowing of audio-visual media (example training packages) and manuals, is also subject to similar procedures. Hardware is currently only lent to academic and general staff members.

Returns are performed in a similar fashion, with the 'labpass' and items retained by the centre and the identification card returned to the student.

Staff and Tutor Borrowings

New tutors and staff members are introduced to the centre by a representative from their department. At this time, all relevant details are entered concerning the person. This information is manually recorded on a Staff Borrowing Form, then entered into the system. Each tutor is identified by a unique two digit barcode which is stored in a folder for quick access. In addition, staff numbers, unique three digit numbers, are stored in the same folder.

Staff members may also be tutors, that is an individual may have two borrowing numbers. A staff member will borrow items useful to their research work, whereas a tutor will borrow teaching sets of software. Delineating loans in this way enables different loan durations to be exercised. ALABS prints out a form letter overnight to staff members with overdue loans.

Class Loans

For a class, tutors will generally borrow one or more class sets of disks. Once the transactions are recorded using ALABS, it is then the tutor or lecturers' responsibility to log the student using a particular software item against the item number on a sheet provided (Class Loan/Borrowing Sheet). At the end of the class, the items are checked against the name of the student so that an audit trail is maintained.

Implications and Future Extensions

Organisational arrangement is as important as a technical solution to the problem of CAL and general course based resource distribution and management. The combination of an appropriate organisational structure and a computer system ALABS has proved to be a reliable solution.

Resource distribution and management in the CAL field, need be no different to that of providing resources for more general computing courses. An appropriate design can satisfy both.

ALABS has proved to be a successful mechanism for the distribution and management of CAL and related items in a tertiary education computing service centre. Successful software for the management and distribution of CAL and related items imbeds the operating procedures of the organisation within the system and provides the minimum cost consistent with the ability to meet the evolving needs of tertiary educational requirements.

ALABS is currently being ported to a PC environment where it will run as Clipper compiled dBase III+ application. Sections of the system have been redesigned to take advantage of the more sophisticated language. The change in language has already resulted in simpler code and fewer databases. Increased speed will result f n the use of a faster host machine. Syst .. performance trials have indicated a conservative 2.2 times speed increase when this port is completed.

Some operational performance bottlenecks exist. Different types of users (tutor class loan/returns, terminal bookings and normal student loans), can lead to servicing delays during peak activity. To correct this, the new version of ALABS will be multi-user, and operate from PCs connected to the LAN. During peak loads two operators might work on the system servicing different user types.

Table 1: Sessional Unsupervised Transactions (1985-1987)

Year	Session	Transactions	Comments
1985	1	1,591	Manually recorded
	2	2,277	Manually recorded
1986	1	3,164	ALABS
	2	5,389	ALABS
1987	1	10,318	ALABS

Ackowledgements

The authors wish to thank Cathy Nicastri for her efforts in testing and managing this system, and Helen Sigalas for her contribution to the implementation of ALABS.

Bibliography

Borovits,I. (1984). *Management of Computer Operations*. Prentice-Hall

Brown,A.R. (1971). *Optimum Packing and Depletion: The Computer in Space- and Resource- Usage Problems*. London: MacDonald

Fried,L. (1979). *Practical Data Processing Management*. Prentice-Hall

Jackson,I.F. (1986). *Corporate Information Management*. Prentice-Hall

Perry,W.E. (1987). *The Information Center*. Prentice-Hall

Sperry. (1985a). *Sperry Personal Computer Microsoftware*. Sperry PC Pilot Support

Sperry. (1985b). *Business Packages for Classroom use*. Sperry PC Pilot Support

Sperry. (1985c). *General Business Programs which operate on the Sperry PC and are useful for classroom instruction*. Sperry PC Pilot Support

Sykes,J. (1987). PCs Present and Accounted For. *PC World* April, 202-208

R . J. Clarke. Previous positions include Computer Operator at the Wollongong Institute of Education Computing Centre, and Tutor for School of Industrial and Administrative Studies, University of Wollongong teaching Microsoft COBOL and BASIC and application packages including Lotus 1-2-3, and Microsoft WORD. Rod designed that schools first External Short Course curriculum, and training notes for SIAS commercial programming and computer applications courses. He has been Operations Supervisor of the Microcomputer Laboratories since 1985. Current research interests include Systems Analysis, and Digital Image Processing. His CAL interests relate to resource distribution and management and PILOT software development and production.

L. Athanasiadis. Support Programmer for the Microcomputer Laboratories since 1985. Apart from ALABS, Louis has written software for a number of educational systems at the Laboratories, including the Educational Testing System for optical mark scanner assessment of secondary school student results.

Bringing the distance student on campus the Riverina way

Brian Cornish, Mark Johnston and Michael Whitelaw
School of Information Studies
Riverina-Murray Institute of Higher Education

The study of computing at tertiary institutions normally involves a substantial component of practical work. This usually takes the form of programming or use of applications software. Traditional approaches to the teaching of computing in the external mode generally use the residential school as the means by which the practical component is satisfied.

This paper describes the use of dial-up facilities as the primary mode of communication and teaching in a post graduate course for students interested in computer applications. The development of software to enhance various aspects of communication is documented. Results of surveys of student attitudes are provided, together with statistics of usage and achievement.

The School of Information Studies at Riverina-Murray Institute of Higher Education commenced the design and development of a new Graduate Diploma in Computer Applications in 1983.

As described in the course proposal documentation (Graduate Diploma in Computer Applications Stage II/III Proposal, Riverina-Murray Institute of Higher Education, May 1985):

> This course is aimed at providing the information management expertise that is necessary for significant computing activity in various professional groups.

The general minimum requirement for admission to the Course will be an Under Graduate Degree from a college of advanced education, university or equivalent institution. However, admission may be considered for an applicant who has demonstrated professional development by one or both of the following:

- Membership of a professional association which requires academic experience based qualifications;
- Professional seniority or status.

Students will not be expected to have any academic knowledge of computing at time of entry. Students who have completed any degree in computing, or an academic major in computing as part of a degree, will not be eligible to enrol in this course.

This course is offered in the external mode; during the development of the course, the School considered various ways of providing computing power for students studying the course without requiring them to attend a large number of residential schools.

The following options were contemplated:

- All the students being required to buy one (1) of a small number of particular brands of microcomputer. for use remotely; or
- All the students being required to buy one (1) of a small number of particular brands of computer and to use direct dial to connect to the Institute's computer; or
- The students being allowed to buy any suitable terminal and use the AUSTPAC system.

The decision was made to use the AUSTPAC facility, and as stated in the course documentation,"the high level of machine excess would normally prove a problem to a course being offered in the external mode, but as the Institute has an AUSTPAC mode, as well as telephone dial-in facilities, students from anywhere in Australia will have access to the Institute equipment, at a cost of something less than $8.00 per hour. Students are required to provide themselves with terminal equipment which will connect to the AUSTPAC facility."

This procedure has proved to be highly satisfactory. Students have used a full range of microcomputers, and some students have used main-frames to connect to the Institute's VAX system. The Institute leaves the responsibility of selecting suitable equipment for connection to the AUSTPAC system to the student.

The students attend two (2) residential schools during the period of this course, one at the beginning of the course, the other at the end. At the residential school at the beginning of the course, the students are introduced, amongst other things, to the AUSTPAC system and taught how to log on to the Institute's VAX system from AUSTPAC.

The first intake into this course was in the Spring Semester 1985, and some of these students recently graduated.

Communication with Distant Students.

AUSTPAC usage

The Institute rents AUSTPAC from Telecom and for the first eighteen months of the course, bore all costs. These averaged approximately $825 per month, being comprised of monthly rental, a fixed charge per call and a variable charge based on volume of traffic and connect time. It then became necessary for the Institute to charge students for use of AUSTPAC over a basic allowance for each subject.

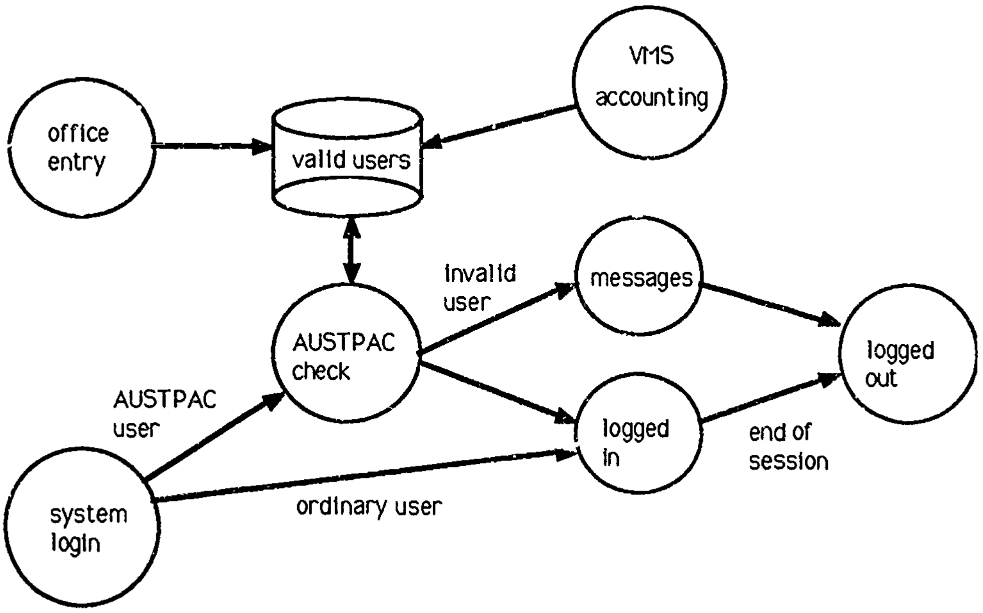


Figure 1: AUSTPAC Accounting system

As Telecom was unable to charge individual users, it was left to the Institute to devise appropriate accounting procedures. VMS identifies users by a terminal number and type, with AUSTPAC users being denoted by an NVA prefix. It thus proved to be a simple matter to identify NVA users at login time and call a program which checks a file of valid users. The file contains the balance of connect time available for use which is displayed on the user's screen at login time. This is automatically updated daily by the VMS accounting facility. Users can thus monitor their usage. Post graduate students receive an automatic allocation of five hours of connect time per subject. Further time is paid in advance through the School office where a program updates the balance in the valid user file. Any user not contained in the valid user file or whose balance is exhausted, receives the appropriate message and is automatically logged out. The overall system is depicted in Figure 1.

Use of electronic mail

The course is structured so that students attend a residential school at the commencement and at the end of their study. At other times the main vehicle for communication is electronic mail via the Institute's VAX computer. Communication, however, encompasses many different aspects of study. Common categories are listed in Table 1 below. Proportions of one lecturer's mail concerning this course are also included.

Initially, electronic mail was used to handle all of the above transactions. The mail facility in VMS version 3 had no sub-directory or folder capability, with the result that mail files were large and difficult to organize. One can easily appreciate that with multiple assignments and 30-50 students, good organization and frequent access to a computer terminal is essential. It also quickly became obvious that users should keep copies of all important mail sent in so that problems or disputes be quickly resolved. VMS mail has provision for the username to which the message is being directed and a subject heading. When receiving mail, the username originating the message and the subject appear when a directory of mail is obtained. Ordinary users of mail tend to be rather vague and casual in the details entered in the subject heading. Typically, they deal with small numbers of messages and may delete them quickly. When lecturing in a course such as this, a much more efficient use of mail is required since one has to glean information as quickly as possible without laboriously reading every message. When sending a message a copy is kept for oneself. Since mail shows the originating username, one soon finds numerous messages with one's own username and no way of distinguishing them short of reading each one. It quickly became apparent that the subject

Table 1 Common categories of communication

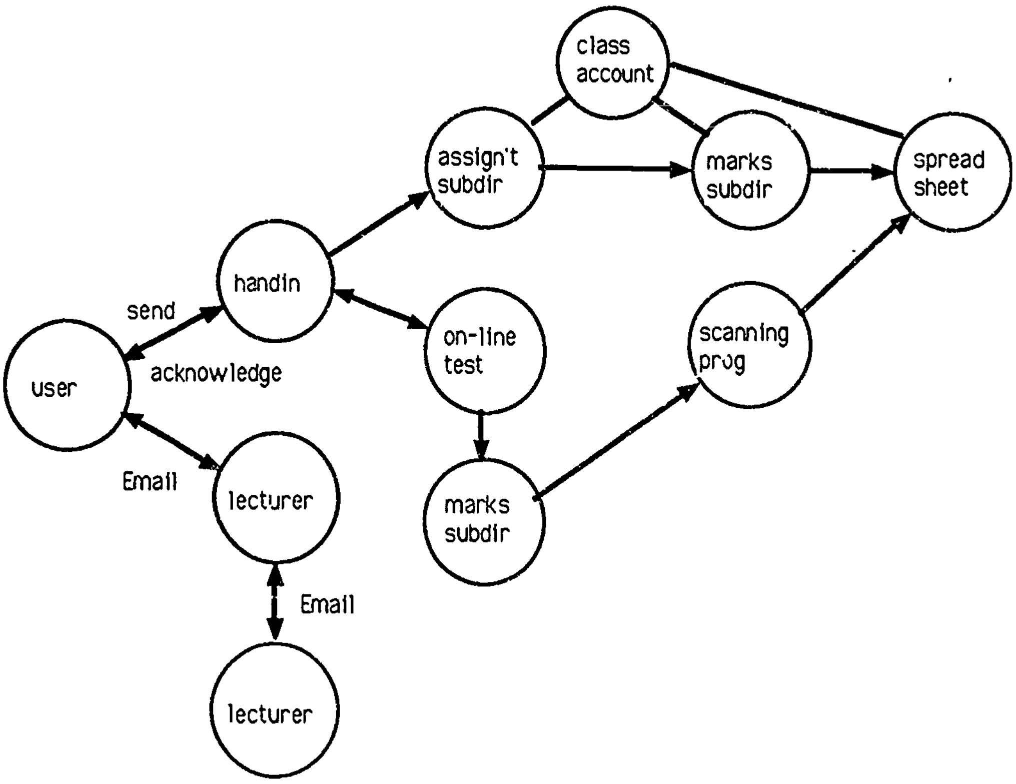
Direction of communication	Nature	% of total mail over 12 months
student -> lecturer	query re coursework	32
lecturer -> student	response to query	29
student -> lecturer	problem report	5
lecturer > student	assignment handout	2
student -> lecturer	assignment handin	1
lecturer -> lecturer	problem report/response	3
lecturer -> student	assignment feedback	28
lecturer -> student	acknowledgement of receipt	N.A.

heading should be used to record the student's username and nature of transaction so that directories of mail could be quickly scanned for information. e.g. Subject DPG1111 Tute7marks

VMS version 4 offered a considerably improved mail facility with provision for separate folders or sub-directories to which mail could be directed. Three folders are normally in operation : NEWMAIL, MAIL and WASTEBASKET. New unread mail appears in the NEWMAIL folder; once read, mail is transferred to the MAIL folder and once deleted, is transferred to the WASTEBASKET from which it can be recovered before the end of that mail session. Other folders can be created to contain the various categories as listed in Table 1. It then becomes a simple matter to scan the directory of a small folder in order to ascertain if a particular student has completed certain work or responded to a request.

The use of mail can be a time consuming activity, especially on a multi-user computer with numerous other demands. A particularly noticeable issue was that of the handling of student assignments. Initially, these were sent to the lecturer concerned or to a class account via mail. The lecturer would then extract each message/assignment, taking care to place it in an appropriate sub-directory using the student's username for later marking. One method which was tried initially was to have students send their assignments to a 'clearing house' account, from where a notification mailing was done. The assignments were then mailed to other marking accounts. This resulted in considerable double handling and inefficient use of time. An on-line

Figure 2: Common communication transactions



menu-driven system of automatic handin.was therefore developed, so as to allow the student to select the subject and the exercise for which he/she wished to submit a file. The file was then copied directly to the appropriate sub-directory of a 'secret' account for later marking. At the same time, a notification of receipt mail message was automatically sent to the student; thus considerable unnecessary handling of mail was eliminated. As can be seen from Table 1, most mail is now confined to student queries and responses.

Some exercises, which are multiple choice and/or short answer in nature, can be automatically marked in the sub-directory to which they are sent. Result files can then be automatically mailed to students and result file directories can be automatically scanned to feed results into a spreadsheet mark book. The automatic handin program has also been adapted to offer an interactive on-line test in some subjects.

A flow diagram representing these transactions is shown in Figure 2.

Student usage

As connect time is limited and since working on-line at 300 baud (the most common rate) is relatively slow, it is important to use time efficiently. Text or data entry is best done off-line on the student's own microcomputer and connect time to the VAX used for uploading, handin or testing. Some subjects, such as the introductory programming subject, enable students to do almost all their work on their own machine. Others, which require special software, make more demands on connect time to the VAX. Some statistics of student usage are presented in Table 2. These were averaged over the months February to June 1987.

From these data it may be reasonably concluded that students are making efficient use of connect time. This may also account for the fact that up to 50 postgraduate students have been able to coexist using only five AUSTPAC lines and three local dial-up lines to the VAX. It can also be seen that most usage is concentrated in the off peak 6 pm - 8 am period.

Student Profile

Some of the characteristics of the first cohort of students were measured. Perhaps the most important characteristic is that of success. Two measures of success are used. The first is quantitative and is the proportion of the course completed in minimum time. The course consists of 6 equally weighted subjects and one double subject. This gives a measure with a scale 0 to 8.

The other measure of success is qualitative; it is the current status of the student. The student may be a graduand , i.e., have completed the course. The student may be continuing. Such a student has taken

Average connect time/session/user	12 minutes
Average connect time/month/user	3 hours
Maximum connect time/month/user	10 hours 30 minutes
Average number of connections/month/user	15
Maximum number of connections/month/user	45
Average total connect time/month	80 hours
Average connect time/month (off-peak cost)	54 hours
Average connect time/month (peak cost)	26 hours

Table 2 Statistics of student usage

		Excluded	Continuing	Graduand	Total
	8			11	11
Proportion	7		1		1
	6				-
of	5				-
	4	1	2		3
Course	3				-
	2	1			1
Completed	1	6			6
	0	5			5
	Total	13	3	11	27

Table 3 - Successfulness

longer than the minimum time to complete the course but still, at the time of writing, may be ultimately successful. Finally a student may be excluded.

Exclusion is not to be interpreted pejoratively. The Institute excludes any students who do not maintain an adequate rate of progress. The student may indeed be attempting subjects and failing them through inadequacy; however, many of the students who decide to withdraw fail to notify the Institute of their intentions. It is only the zero rate of progress that enables the Institute to recognize the withdrawal and remove the student from the course. Unfortunately, the number of students who withdraw and are polite enough to inform the Institute are sufficiently small that it has not been felt necessary to create a special category.

Those first cohort students who completed the course in minimum time are now graduands. Table 3 shows the distribution of the students between the two measures of success.

Current graduation rate is 41% and could rise to 52%. Most students who do not succeed, fail in one or both of the first two subjects studied. Indeed 85% of the students now excluded had been excluded by the end of the first programming subject. This makes early performance a good predictor of ultimate success.

Some other predictors that could be used are verbal ability, spatial ability and attitude. These predictors have the advantage of requiring only a short interview to be assessed, so they are a tempting screen for administrators of a popular course who have to decide who is to be admitted to a course. These predictors were measured at the start of the course and compared against final success. As will be shown below, none were found to be as satisfactory a measure as success in the first subjects.

The test used for verbal and spatial ability was Heim's AH5 Test (Heim 1968) of group intelligence. This test is a group intelligence test designed to assess people already known to be towards the top end of other intelligence scales. Results have been related to University and Teachers' College students. The test is a relative test, intended to disperse the group across the scale rather than measure ability in absolute terms.

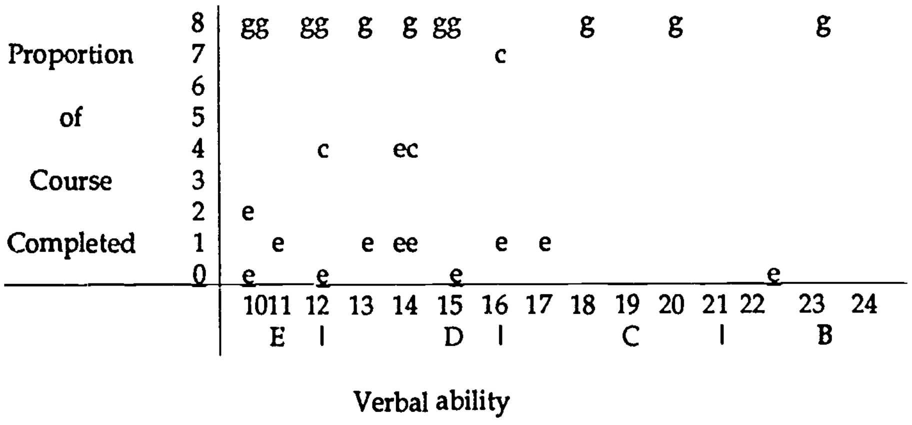


Table 4 - Successfulness versus Verbal Ability (one student did not do the test)

For ease of comprehension, scores are converted to grades such that the first 10% are given an A, the next 20% are given a B, the middle 40% a C, the following 20% a D and the lowest 10% an E. Analysis of the test was carried out by Reid-Smith(1985).

The first part measures verbal ability. The students scored between 10 and 24. This spreads the students between E and B at University level; or E and A at Teachers' College level. Table4 provides the distribution of verbal ability versus achievement level. A student is recorded as an "e" if he is excluded, a "c" if continuing and a "g" if a graduand.

It is clear from Table4 that success does not depend on verbal ability. It must be remembered that the population is already pre-selected so such a statement may not be true of the full range of verbal ability.

The second component of the AH5 test used was that which measured spatial ability. Again, it is a test intended to show differences in a narrow range of abilities rather than measuring absolute abilities. The results are in Table 5.

There is no apparent relationship between spatial skills and success shown in Table 5.

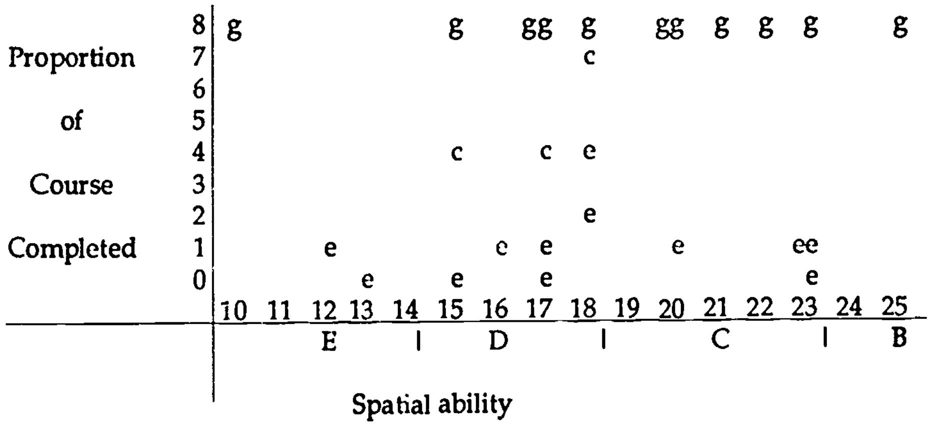


Table 5 - Successfulness versus Spatial Ability (one student did not do the test)

```
             8 | g                   gg      g   g   g       g       gg  g   g
Proportion   7 |     c
             6 |
    of       5 |
             4 |     c       c                               e
  Course     3 |
             2 |                                     e
Completed    1 |     e       e   e   ee                      e
             0 |         e               e   e                               e
               ----------------------------------------------------------------
                 26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41

                 Attitude
```

Table 6 - Success versus Attitude.

In fact, there may be some relationship between very poor spatial ability and failure, as the student who performed badly on the spatial test but graduated was sick on the day of the test. Her subjective impression was that her performance was significantly affected.

Another characteristic measured was that of attitude on entry. The instrument used to measure attitude was a subset of the questionnaire used by Wilson and Braun to distinguish between groups of students (Wilson and Braun,1985). The subset were those questions in the original questionnaire found to show significant differences in attitude towards computers. On analysis, Reid-Smith showed this subset to be 88% perfect in being uni-dimensional with a neutral point at 33 (Reid-Smith, op cit).

The attitude of students on entry is measured against success in Table 6. While it is clear that a neutral to positive attitude helps, it is still possible for a student to graduate in minimum time even if a rather antagonistic attitude towards computers is held on entry to the course.

Conclusions

AUSTPAC has proved to be a very satisfactory method for allowing Distant Students to communicate with their lecturers and to use the Institute's VAX computers.

From the results of the four measurements used with the first cohort of students in the Graduate Diploma, it would appear that the probability of success is enhanced by a good attitude towards computers, is not limited by spatial abilities, is even less affected by verbal abilities, and is best predicted by the results in the initial subjects.

References

Reid-Smith, E. R. (1985). *Some Attitudes and Skills of Computing Students: An Analysis of a Survey undertaken by M Whitelaw at RMIHE*, Internal report.

Heim, A.W. (1968). *AH5 Group Test of High-Grade Intelligence: manual.* Slough (England): National Foundation for Education Research, r. e. [Restricted Test]

Wilson, J. D., Braun, G. F. (1985). *Psychological Differences in University Computer Student Populations.* ACM SIGCSE Bulletin, . 5. 166-177

The three authors are members of the School of Information Studies, Riverina-Murray Institute of Higher Education. Brian Cornish is the Dean of the School and as such, is responsible for course development. He has had many years experience in computing, commencing on Silliac in 1960. His research interests include effective distance education teaching techniques. Mark Johnston is a lecturer in the School and his recent teaching experience includes introductory programming subjects in the Post Graduate Diploma, and robotics and expert systems in the degree course. Current research interests include robotics, expert systems and the applications of computers in astronomy. Michael Whitelaw is also a lecturer in the School and his recent teaching experience includes algorithms and languages, business programming and data structures subjects at both graduate and undergraduate level. His research interests include automated warehousing and effective teaching techniques.

Computerised formative assessment — A powerful new tool

Peter Fleming
Director of Educational Development
Family Medicine Programme

The Royal Australian College of General Practitioners has some 2000 young doctors undertaking post graduate training in the family medicine programme. They are scattered all over Australia in supervised practical experience and supplying them with high quality educational materials is a major problem. An interactive computerised, formative assessment system has been established on Viatel called CHECKUP (Computerised Home Evaluation of Clinical Knowledge, Understanding and Problem solving) .It consists of a bank of over 8,000 TF questions and a series of diagnostic and management problem solving exercises. The CHECKUP software runs on a VAX and is available for CAEs, universities and continuing education programmes for almost any subject area.

Most teachers are concerned about making their teaching more effective and efficient. The best way to do this is for your students' learning to be more effective and efficient and the best way to do that is to ensure high levels of motivation to learn and to provide feedback for them on their learning.

There are many ways that feedback can be provided to students. One such way is to give them access to assessment processes during their learning that reveal where they are going right and where they are going wrong, and why. This type of assessment is called formative assessment and aims to improve performance by providing feedback. The more common - summative - assessment usually takes place at the end of learning, is designed to grade students, and generally provides little feedback on their specific strengths and weaknesses.

It is now possible to provide computerised formative assessment to students, in the area of knowledge and problem solving, using software written for this purpose called FACTS (Formative Assessment Computerised Testing Service). FACTS is written in Fortran and was developed by the RACGP's Family Medicine Programme for their 2,000 or so young doctors in an apprenticeship—based, four year post—intern programme in hospitals and practices throughout Australia.

The RACGP/Family Medicine Programme call their medical version of FACTS CHECKUP (Computerised Home Evaluation of Clinical Knowledge, Understanding and Problem solving). Delivery of CHECKUP is through Telecom's Viatel system which provides ready access to the computer from anywhere in Australia via the public telephone network at a reasonable price (9 cents a minute in business hours and 6 cents a minute outside).

The Family Medicine Programme currently has 50 Viatel terminals circulating among its trainees, so that each trainee should have access to CHECKUP at least once a year. Advertising on CHECKUP supports most of the operating costs.

FACTS consists of two main parts:

- a bank of true/false questions designed to assess factual knowledge
- a bank of diagnostic problems designed to assess problem solving ability.

Question bank

The question bank is designed to be accessed in two different ways. A menu structure allows the user to focus on an area of interest or the computer will select questions at random from a "pool" of questions. For example, the CHECKUP system is arranged to allow access via:

- Diseases and Conditions,
- Random,
- Signs, Symptoms and Abnormal Tests,
- Special Subjects, and
- Systems,

where Random and Systems are pools. Questions are entered in several parts of the bank; for example, all questions go into the random pool, while cardiac questions will also be in the Circulatory System pool as well as in part of Diseases and Conditions and perhaps in Signs and Symptoms or Special Subjects. A summary of the CHECKUP menus is attached.

The True/False questions are arranged in sets of four parallel questions a,b,c,d. If the user answers 1a correctly, 2a is presented, but if it is answered incorrectly, 1b is presented followed by 1c and 1d. This is to test whether a small slip has been made or a significant lack of knowledge has been identified. If these additional questions are all answered correctly, 2a is then presented and no "error" is recorded. If, however, one or more of them is answered incorrectly, the computer records this topic as one needing attention. Likewise if three or more of the questions in one topic are answered incorrectly, but all the b, c, and d questions following are answered correctly, this is also considered an area requiring attention. Arranging the questions in sets of 4 also has the advantage of being able to present a fresh set of questions each time the user re—enters a topic (up to 4 times).

At the end of a session, a list of topics undertaken is presented, along with a list of those topics requiring further attention. This is provided (if required) as a printout to both the FMP doctor and his/her training adviser for discussion at their next meeting. No score is calculated, as the emphasis is on remediation rather than ranking.

The system also allows users to send messages to FMP at the end of a session, e.g., complaining about the accuracy of a question or the ambiguity or relevance of a question. This is an important feedback mechanism for maintaining the bank in a healthy state.

Statistics are collected on the use of the system which will allow comparisons to be made between performance of different groups of doctors and it also identifies questions with high failure rates. These may indicate poor questions or significant areas of deficiency in doctors' knowledge which would then need to be addressed through educational channels.

Problem bank

The problem bank is designed to give feedback on the learner's problem solving strategies. It does this by presenting the initial data on the problem and then allowing the learner to ask questions in a number of categories. The format is open and the learner can return to any category as often as he likes and ask as many questions as needed. When the solution has been entered, feedback is provided.

The CHECKUP problems provide an opening description (the patient's presenting symptoms) followed by a pathway frame which offers alternative types of action which could be taken (history, examination, investigations, emergency treatment, etc.). This pathway frame can be returned to at any time to select a different type of action.

Once a particular type of action is selected, the user specifies what he wants by typing in the first three letters of the specific action or information required (e.g.,DIE for Diet if you want to know about the patient's diet or SMO for Smoking History or CAT for CAT scan). The computer then provides a menu of all items in that section starting with three letters, or, if there is only one item, it supplies the response.

The user works his way through the problem until he thinks he has solved it, when he enters his solution. The computer then gives the expected solution plus a description of how the problem could have been solved. This is followed by feedback to the user in the form of a list of items chosen which would have been useful in solving the problem, marked + ve, but also includes those which would have been counterproductive (in CHECKUP this includes unnecessarily invasive, harmful or costly investigations) marked —ve. These enable the user to compare his list with a similar one for each of two competent professionals who did the same problem. It is illuminating to see which questions they asked and to realise that even experts make mistakes occasionally!

The problems can be scored; in fact this system was used by the RACGP candidates sitting for the Patient Management Problem component of their Fellowship examination this year in all the capital cities in Australia.

The exam problem choices are printed out in chronological order. This feature can be used to follow the thinking of the person solving the problem. This process is helped if the users are asked to write down their hypotheses and indicate which hypothesis they are testing and whether the response confirms or refutes the hypothesis.

Facts in tertiary education

Universities and CAEs could provide a valuable and powerful learning tool to their students by establishing FACTS on their computers and encouraging departments and schools to enter their own questions and problem banks in the system. This could lead to a substantial increase in the effectiveness and efficiency of their teaching programmes, at least in the area of knowledge and problem solving. The pooling of questions and problems by all institutions around Australia which conduct courses in the same area, e.g., nursing, medicine, chemistry or botany, would create a magnificent data base for each course to use.

Other developments

The RACGP Family Medicine Programme is also providing a management problem on Viatel; here the diagnosis is given, questions are asked about management, and feedback is given on the choices made. This is a branching programme originally designed by Monash University's Department of Community Medicine and based on the Modified Essay Question (MEQ) format, which does not require interactive capability and so is provided on normal Viatel frames.

A Special Interest Group (S.I.G.) has been set up on Viatel called General Practitioner Network. This is in effect an electronic scribble board for GPs on which they can put notices about clinical matters, conferences, interesting articles, useful resources, etc. Notices drop off automatically after a nominated time (14 days in this case) or can be removed by their owner. This S.I.G. will help to break down the professional isolation of many GPs.
A read only bulletin—board has also been established with dates and times of meetings, conferences, etc, and important information.

The future

An update service will be added to CHECKUP which will allow the doctor to read the latest developments on, say, AIDS. This should prove a very valuable service, but requires substantial work to establish initially and to maintain at a high standard.

A community Health Information Service is proposed which would be open to the general public and enable them to access health information and possibly pay for the time on the system through coin—in—the—slot terminals in public places such as pharmacies, hospitals, libraries and schools. If successful, this would allow most Australians to access high quality health information at a reasonable cost. Access will probably be via menus of diseases and conditions or problems which list information under headings such as:

- what causes it?
- what are the symptoms?
- how can I prevent it?
- what treatments are available?
- can it be cured?

It is expected that other professional groups will establish data bases on Viatel similar to CHECKUP to provide continuing education for their members in the field.

For further information,

- about the education aspects, contact:
  CHECKUP COORDINATOR
  Family Medicine Programme,
  70 Jolimont Street,
  JOLIMONT, 3002.

- about the software, contact:
  Merlin Communications,
  147 Eastern Road,
  SOUTH MELBOURNE, 3205.

Acknowledgements

The FMP development team consists of: Ms. Jude Blackwood, Computer Services Co-ordinator, Dr. Wes Fabb, Director of Education, Dr. Peter Fleming, Director of Educational Development, Dr. Jack Marshall, Director of Educational Research and Resource Development.

Computer enhanced distance learning

S. Latham, W. Moore, G. Ritchie, B. Rothwell and L. Wilde
Mitchell College

This paper presents preliminary results and suggestions on the use of micro-computer/mainframe linkages in the teaching of undergraduate courses. The findings presented are the result of a co-operative effort between Mitchell College of Advanced Education and IBM Australia. Concrete data is given on the use of microcomputers in the study of a computing subject and a social science subject for first year undergraduates at Mitchell College. Various preliminary findings based on computer usage data and information from questionnaires distributed to the participants is presented and appropriate conclusions drawn.

Mitchell College of Advanced Education is a College that has the majority of its students studying in the external mode. The external mode of teaching means, in Mitchell's case, an extension of the New England model as distinct from the correspondence model. The student is required to attend a four or five day residential school at the College in the middle of each semester as well as optionally attending some weekend schools presented at various centres. The student receives a set of study materials containing assignments and unit information, a study guide and a booklet containing selected readings. Other material such as cassette recordings and videotapes may also be made available.

The students studying in this mode are distributed over a wide geographical base. The units in various courses studied by external students range from business and law through to computing, social science and applied science. A number of units studied in the courses provided by Mitchell College require the use of some form of computing. This usage is split into two areas at Mitchell College; that of units 'about' computers and units 'using' computers. The latter units typically would use a computer to run statistical and word processing packages and the former units are those using the computer to compile and execute programs in computing courses.

Much effort is expended at Mitchell College by lecturers and external studies staff in preparing the study guides used by the students. The staff at Mitchell are constantly seeking new methods to enhance distance learning. One of the major enhancements addressed in this project is to increase the lecturer/student, student/student and student/institution interaction.

Microcomputer Assisted Distance Learning

The main driving force in improving distance education has been the development of new technology. The application of new technology to distance learning has received a great deal of attention and is reported in the literature (Bates, 1984).

The use of microcomputers as a delivery system for material, linked to computer(s) at a central location has been discussed but not as yet implemented (Clark, 1986). Attempts have been made, for example, by Capricornia Institute of Advanced Education, to use computers in distance education. This attempt ,however, was to use microcomputers at study centres and not in individual students' houses (Cook, 1985). Some tertiary institutions, Mitchell College amongst them, have encouraged students who have microcomputers at home to use a communication program and modem to dial into the institution's mainframe and to use the facilities to complete assignments and use any other facilities that may be available. This method of use has been encouraged but not investigated along any cohesive development plan.

A pilot project using a microcomputer/ local area network/mainframe linkage was initiated at Mitchell College in the latter half of 1986, with the co-operation of IBM Australia.

Project Aim

The aims of this project fell into three main categories: technical, student background and educational enhancement. Specifically the aims were:-

1. to investigate whether the use of a micro- computer at home would enhance the students' learning process;

2. to investigate the technical and organisational aspects of establishing a reliable communication network between Mitchell College and external students for use in distance education; and

3. to investigate the suitability of different types of unit presentation and content coupled with the students' background in using computing facilities.

One of the main incentives for these objectives was the lack of published experimental information of any microcomputer/ mainframe studies in distance education. Most information in this area is of an anecdotal nature. It was decided at Mitchell to embark on a pilot study to collect and publish some information in this area.

Project Hardware

The facilities needed to implement the project have been provided by a co-operative venture between IBM Australia and Mitchell College. IBM Australia lent a mainframe and associated equipment and Mitchell College supplied microcomputing facilities to students participating in the project.

The mainframe equipment used was an IBM 4361, a console, 4 colour terminals, a line printer, a tape drive and 1.2 Gbytes of disk storage. The value of this equipment is estimated by IBM to be of the order of $750 000. The operating system used for the project was MUSIC running under the IBM VM operating system. IBM also provided Pascal, COBOL and FORTRAN compilers as well as a graphics package, a communications utility called PCWS and a package for the implementation of computer aided instruction material.

Mitchell College supplied each of the 30 students participating in the project with an IBM JX personal computer, colour monitor, two floppy disk drives and a Netcomm 1200A Smartmodem. Each student was supplied with a copy of a wordprocessing package, the MS-DOS operating system and utilities and a copy of PCWS. PCWS enabled communication using IBM 3270 emulation between the microcomputers at home and the IBM 4361 mainframe at Mitchell College. The use of a synchronous protocol emulation at home is a significant feature of this project. AUSTPAC communication was installed to the ethernet Local Area Network (LAN) that is used at Mitchell College. The students dial into AUSTPAC from home and are connected to the Col LAN. Once connected to the LAN, provision for further connection to any of the host systems can be established using a simple LAN menu facility. The IBM system is one of the host systems on the LAN.

Project Software

The key piece of software for the project was a microcomputer program running under the MS-DOS operating system called PCWS.

The PCWS program allows the student to use the IBM JX at home as if using one of the colour terminals connected to the mainframe at College. The PCWS program also allows the student to transfer programs/assignments from/to the mainframe in a very convenient fashion. This is achieved due to the integration of PCWS and the MUSIC operating system supplied by IBM. This operating system was developed at Magill University in Canada and was designed to be simple to use and yet provide a powerful working environment. A wordprocessing package, PC Write, from the public domain was also supplied to students allowing them to wordprocess assignments on the IBM JX machines for later transfer to Mitchell College.

Project Organisation and Implementation

Thirty students were selected for participation in the project. These students were drawn from two units of work; Computing I and Methods and Statistics in Social Science. The first unit is from the Industrial Mathematics and Computing degree course at Mitchell and the second is a unit in the Social Science degree course at Mitchell. Both units are first semester units. The project had two groups of 15 students, one studying the social sciences and the other studying the introductory computing subject that required them to learn Pascal programming techniques.

These students were asked to attend a day long briefing session at the beginning of autumn semester 1987. During this session they were to receive their IBM JX microcomputers and also learn how to use the following:-

1. PCWS (Communications program)
2. Modem connection
3. The IBM MUSIC editor and environment
4. PC-Write (Word processing program).

The students were issued with a manual detailing function key sequences, Austpac connections, a tutorial (brief) on using PC-Write and other technical information needed to use the complete system. The two groups of students were briefed on two different occasions at Mitchell College.

The students were also given a network user identification (NUI). This is used for accounting with the Austpac system. The students were trusted not to abuse the use of the NUI. The College paid the bills that accrued through the use of the two NUIs that were given out one to each group of students. The students only had to pay for a local call when using the Austpac NUI. This is not necessarily a definitive arrangement and will change according to circumstances.

Study Profile

The two sets of students had different study needs from the project. The social science students were studying a unit that was split into two distinct sections over the semester. The first section required them to learn some basic statistics for later application to social science problems. The later section consisted of essay type assignments on various social science topics. These students were able to use an elementary

CAL program on the mainframe to revise and learn some elementary mathematics work as well as participating in a regular conferencing session every 2-3 weeks.

At the beginning of these sessions, the lecturer would pose a controversial topic and invite discussion on the topic using electronic memo techniques from the participating students. Both of these uses of the mainframe were voluntary and were not part of the required coursework of the unit. Students were also encouraged to use the PC-Write package at home to wordprocess assignments used in the latter part of the unit and then to electronically transmit them to College for marking.

The computing students on the other hand were required as part of their unit to complete 8 assignments that required the application of the Pascal programming language. The first assignment was a written one, but the other 7 assignments needed to be run and developed on a computer. For these students an IBM Pascal compiler was available on the IBM 4361 as well as the required data sets to be used in each of the assignments. These students were to use the computer as an integral part of their assessable work in the computing unit.

All students were able to make use of the electronic mail facilities to leave or read memos for lecturers or other students. Another facility that was available was the access to library facilities using the CLANN library system. This system allows access to a database of library titles and topics from various participating libraries. Mitchell students are able to request these books as required.

Results

At the end of the autumn semester 1987, a questionnaire was sent to all participating students. An additional questionnaire was sent to the social science students concerning topics that applied only to these students.

Preliminary results from this project can be broken into two groups: technical and educational.

Computer Usage and Empirical Results

The project had as one of its aims the investigation of the technical feasibility of establishing a micro/mainframe link and to gather some facts on the establishment and running of such a link. After an initial time in early March 1987, most of the students had mastered the technical problems of establishing a usable linkage from their homes to the mainframe at Mitchell College. From answers in the questionnaire, the average time to become familiar with the equipment was 2-3 weeks.

From the 16th March 1987, records were kept of machine usage on the IBM. The accounting files were processed using some application programs and relational database techniques to produce a file with only student logon information. This consisted of a relational table with the attributes of logon identification, type of student, connect time, connect port, time of connection, date of connection and an IBM IPL-identification value. From this table three figures of interest arise.

Figure 1 shows the hourly usage of the 4361 during the day. The results show what would normally be expected, that is, most students use the machine from early afternoon through to approximately 2300 hours. It should be remarked that this figure includes weekends, which partly explains the low usage of the computer link during working hours.

Figure 2 shows the usage by day of the week. Monday is taken as day 1 and Sunday as day 7. Again the usage is as expected, Saturday, Sunday, Monday and

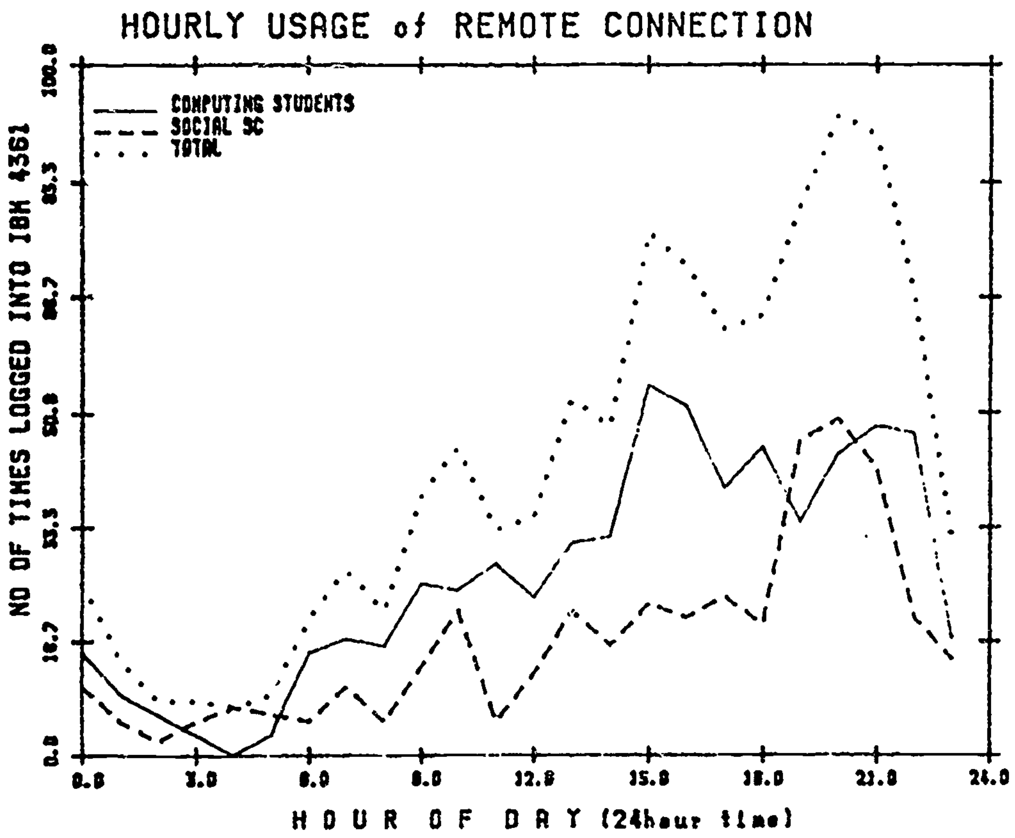


Figure 1: A graph of the hourly usage of the IBM 4361 connected to the LAN at Mitchell College

Tuesday are the heavy usage days. The last figure shows the connect time and the day in semester. From the graph it can be seen that the usage by the social science students tailors off as the semester progresses. This may reflect the approach of exams and the lack of any real need for the computer except for transmission of asisgnment material.

It is interesting to note that usage of the machine was minimal over the Easter breal (days 20,21,22 on the graph). The usage of the machine by the computing students is again what might have been anticipated: sharp peaks of usage when assignments are due! All the figures provide some empirical data on micro/mainframe usage. The data does not include access to the CLANN library system or development work done on the microcomputer at the students' homes. The total usage of the system is given in Table 1.

Table 1	Total Hours	Average / logon day
Computing stud.	304.62	4.35
Social Science	134.82	2.32
All Students	439.44	6.01

Column two of Table 1 shows the average con.puting time per day for the indicated students.

Educational Results

The results presented here are a result of analysing the questionnaire and other re-

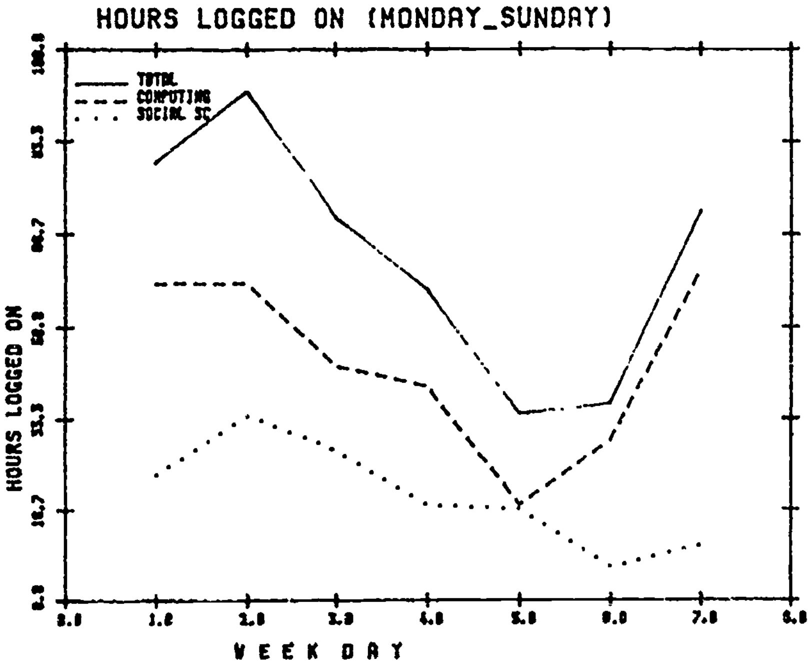


Figure 2: A graph showing the usage of the IBM 4361 connected to the LAN at Mitchell College for each day of the week.

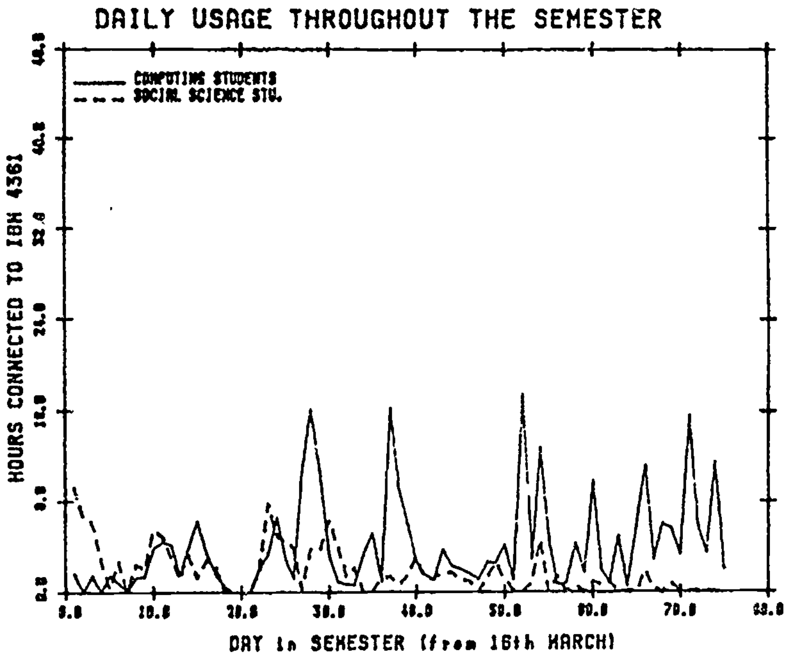


Figure 3: Graph showing the usage of the IBM 4361 connected to the LAN at Mitchell College for each day of the Autumn Semester starting at 16th March.

sponses. A selection of questionnaire results are presented in brief form in Figure 4. Students were given a scale of 1-10 from which they could reply to questions on the questionnaire form. Students answering in the range 4-6 were put into the unsure category in Figure 4.

From the results given in Figure 4 it is easy to see that the students generally welcomed the use of the microcomputer in their studies at Mitchell. Some responses reflected distinctive biases from each of the selected student groups. This difference is shown quite clearly in the response given in question 11 (interaction with other students). In the social science group, 6 of the 9 responses agreed that they benefitted quite markedly from interacting via electronic memo with other students. In contrast, the computing students stated that they did not use the opportunity to send/get memos from other students very much and it was not all that helpful. At this stage we can only conjecture at the reasons for this response. It is worth noting, in connection with the results from question 11, that all the students indicated that the electronic memo facilities were relatively easy to use and no clear bias was seen from either group on this question.

Another topic that showed a clear bias between the two groups was that of benefit of the facilities in doing assignments. Eight of the nine responses from computing students agreed that they had a definite advantage over other students in their unit by using the microcomputer/mainframe link. The explanation for this seems to be that doing computer assignments interactively is far superior than doing them via Australia Post. It is suprising that the social science students did not think that they had an advantage over other students in the unit.

Questionnaire Responses

All the students agreed that more time should have been allowed in getting them acquainted to using the computing link. This area of the project can be improved using past experiences and having learnt what are the common mistakes and needs of the beginning student. Improvements have been made as the project progresses. One such improvement was the writing of a simple logon procedure that made dialling up and logging on much easier.

	QUESTION	yes	no	unsure
1	Computer link reliable	11	3	4
2	Computer link was easy to use	13	4	1
3	Confident using the link	13	2	3
4	Computer at home was helpful in studies	15	2	1
5	Linkage useful in contacting lecturer	12	3	3
6	Definite advantage over other students	10	6	2
7	I was adequately prepared to use the equipment	4	9	5
8	More time was needed to repare me	17	1	-
9	I will obtain the use of a micro in the future	15	2	1
10	Assignments were done more efficiently	13	3	2
11	Benefitted using elect. memo with other students	8	8	2
12	Electronic memo was easy to use	14	3	1

Figure 4: Summary of some questions asked of the project students at the end of the semester.

The social science students responded that they spent less than 12% of their time using the remedial mathematics program (MATHE) that was set up using a CAL package supplied by IBM. This was quite a surprising result.

The conferencing sessions was another area where the usage by the students was very disappointing (15% of their time).

The use of the wordprocessing package was rated very highly by the social science students. Most of the time was spent in sending assignments to College (J 20% of their time) and in sending/reading electronic mail (J 30% of their time)

The final grades obtained from exams by students in both groups was of a high standard (a number of distinctions and A grades), but no significant results can be reported due to the project selection criteria not being completely random and the small sample size. However, the general level of grades gave an indication that the use of the micro/mainframe link was beneficial and the use of a more rigorous experiment might perhaps be able to determine whether students' grades are influenced by having access to a computer linkage as described in this paper.

Another interesting, although not unanticipated, result emerged from the computing students. This was the abandoning of the use of the IBM Pascal compiler to use the Borland product, Turbo Pascal. This was because of the ease of use of this product compared to debugging a program over a phone line.

There was a general feeling amongst all of the students to have a small printer available with the microcomputer.

The majority of students (95%) stated quite strongly that the use of electronic mail in contacting lecturers and other students was their greatest benefit. This contact with the College and having almost immediate response made the students feel "as if we are part of the student body". The use of electronic mail seemed to reduce the sense of isolation for students studying in the external mode and made some of them feel that they were not being doubly disadvantaged by not having access to large city libraries and postal services.

Conclusion

The project has provided Mitchell College with some valuable experiences in organising and incorporating the use of microcomputers in the teaching of some external units.

The preliminary results of this pilot project indicate that the students do benefit from the use of a microcomputer at home linked up to a mainframe at College, this linkage being an integral part of a students' studies. The use of electronic mail as part of a unit of work has proved to be of benefit to students and staff. However, no conclusions can be drawn from this pilot study about the ramifications of such computer technology on the final exam results.

The second goal of the project, that of investigating the feasibilty of establishing a reliable computer linkage, has shown very promising results. The establishment and maintenance of a reliable, remote computing environment for students throughout the semester, has proved to be one of the successes of this pilot project. It has been demonstrated that a section of studies at Mitchell College can be enhanced and have incorporated successfully into it a microcomputer/mainframe linkage that provides students with a better means of communication and study than that which presently exists.

The project has given some preliminary facts on which to base any further work. The project has established an organisational and educational framework for future developments in the use of computer enhanced distance learning. It remains to be seen how other units of study can be adapted to exploit this particular technology.

References

Bates, A. W. (ed) (1984) *The Role of Technology in Distance Education*, New York, St. Martins Trees.

Clarke, J. A. (1986) Some Reflections on CAL at the Tertiary Level in Australia, *Proceedings of the Fourth Annual Com-*

puter Assisted Learning in Tertiary Education Conference, Adelaide 1986, 64-71.

Cook, L. G. (1985) Considerations From an Attempt at Implementing CAL in Distance Education, *Proceedings of the Third Annual Computer Assisted Learning in Tertiary Education Conference,* Melbourne.

Contact Author:
Dr. W. E. Moore
School of Applied Science
Mitchell College BATHURST 2795

Dr. W Moore and Dr. G Ritchie are lecturers in computing and social science respectively and have been responsible for developing the unit content and evaluation of the IBM project at Mitchell College. Dr. B Rothwell is the Deputy Principal at Mitchell and is the director of the IBM project. Technical and administrative expertise has been supplied by Mrs. S Latham and Mr L Wilde. Mr L Wilde has been the operator/systems programmer for the whole project.

Quality distance education = computer based feedback + electronic mail

Donald Munro
School of Information Studies
Riverina-Murray Institute of Higher Education

The Riverina-Murray Institute of Higher Education offers a Post-graduate Diploma in Computer Applications. The course is offered only externally and utilizes AUSTPAC as the prime instructional and communication medium. The paper describes the types of facilities which have been investigated and developed, in an attempt to provide an enhanced learning environment for the external student. Two of the facilities discussed are the use of ELECTRONIC MAIL, and COMPUTER BASED FEEDBACK. The use of Electronic Mail has meant that the external student can have the same types of regular communication with lecturers which an on-campus student takes for granted. Computer Based Feedback has provided a periodic assessment mechanism, controlled by the student, which highlights areas of learning difficulty by directing the student to material in the prescribed texts and notes. The combination of these facilities has proved highly effective. Not only has the marking load on staff resources dropped, but students are able to ask questions relating to their particular difficulties, and receive direct advice on overcoming those problems.

A lot has been said and written about the benefits which could be achieved in distance education by the utilization of modern communications technology. At the Riverina-Murray Institute of Higher Education in Wagga Wagga, the approach adopted by the School of Information Studies has been typically careful and low key, with the Institute proposing a Post-graduate Diploma in Computer Applications in early 1985; the Diploma commenced with an initial intake of 28 students in June of that year. The Course was distinctively unique in that it was offered only externally and utilized AUSTPAC as the prime instructional and communication medium. Students were to purchase and install terminal equipment in their place of study. Now, two and a half years later, several salient features of this style of education have emerged which are worthy of mention.

The Institute is no stranger to the task of teaching computing. In 1979 an Associate Diploma in Computing was introduced, in the full-time, part-time and external studies modes. This course has always filled its quota of student places, and it was the persistent demand in the external mode from applicants already possessing an academic qualification which led to the development of the post-graduate offering. In the last five years, pressure from external students possessing their own micro-computers encouraged the Institute to install three DIAL-UP lines to the Academic VAX 11/780 and 8200 computers. This facility allowed the preparation of programs and data off-line, followed by concentrated periods of transmission during the STD off-peak periods. Students utilizing these facilities made generally better use of the time available to them during residential schools, learned more and achieved better academic results. In short, a higher quality of distance education was being achieved through little more than providing appropriate hardware.

The introduction by Telecom of the AUSTPAC Communication facility presented a challenge to the computing lecturers:

With the removal of punitive STD charges, to what extent could extensive access to existing and newly developed software packages, support and enhance the learning opportunities for the External Student?

The Post-graduate Diploma in Computer Applications concentrates on six major aspects of computing. The student is provided with: a comprehensive programming background; a solid understanding of computer management; a good knowledge of fourth generation facilities; the fundamentals of data base design and data communications, and experience using Structured Analysis Techniques. A final individual case study major project draws together the student's newly acquired knowledge and experiences.

The course, which consists of two years part-time external study, commences in June with a five day residential school. Unlike residential schools in the Associate Diploma, where the emphasis is centred around practical tuition and computer access, in this residential school the Post-graduate Diploma students are oriented to the Institute and its expectations of them, the equipment which they will be accessing from their remote locations and the lecturers they will be interacting with (sight unseen) for the following two years. Examples of appropriate terminal equipment and the use of AUSTPAC are also covered, together with a general introduction to LOGGING-ON, and using the HELP and electronic MAIL facilities.

The distinguishing feature of this course is the use of the AUSTPAC communication facility as the primary communications link between the student and the lecturers involved.

As these students have no extensive background knowledge of computers (there is no pre-requisite computing knowledge), the residential school also allows inspection of the VAX computing facilities, so that students have an idea of the location of the machinery with which they will be interacting. This is an important concept to establish, as most of the students' home terminals will be micro-computers with their own processing and storage facilities. We wish to distinguish between "local" and "host" storage and processing, and the various communications between them.

Student equipment

During the design of the course, the Institute lecturers looked at the advantages which would accrue from definitively specifying the equipment which the student had to acquire. Standardization would allow the preparation of programs which could be "down-loaded" and RUN in LOCAL mode, and would allow the exploitation of graphics and colour. In addition, the equipment could be completely checked out and the support effort would be reduced. On the other hand, many potential students already had computer equipment which would be suitable for textual transmission, and regardless of which equipment was standardized upon, it would be obsolete in a relatively short period of time. The decision to remain centred on the Institute VAXs was made on the basis of:

- the relatively small amount of graphics and colour related instruction
- insufficient staff to develop a new set of programs centred around a particular micro-computer
- the existing set of programs used in the Associate Diploma and based on the VAX being seen as providing a good base of appropriate material for the Post-graduate Diploma students.

The introductory programming subject (a full year subject) is taught using the BASIC language. Other languages were considered. However, as most of the micro-computers have BASIC built-in, and as the Associate Diploma course already used BASIC in its introductory subjects, any move to another language without significant advantages would have produced an unacceptable strain on staff resources. The pedagogical advantages of being able to test and run small programs in LOCAL mode was seen to be a great advantage, particularly when the same program could be uploaded and then run on the VAX.

This approach works well because the initial programming instruction concentrates on a STANDARD SUBSET of BASIC statements which are compatible across a wide range of equipment, rather then exploring the richness of various BASIC dialects. In most cases initial programs ran in LOCAL and HOST without change, and more advanced programs required only a few global changes using the EDITOR (which had been taught by that stage). Another advantage of this approach is that the student develops a respect for the power and flexibility of the micro being used; the "even my own machine can do it" syndrome. In fact some students took great delight (and spent many hours) converting advanced examples provided on the VAX into versions which would run on their own machines. The value of this to the educational process is self-evident.

Electronic Mail

Most exercises and assignments are submitted on the VAX by electronic mail. Using this approach, the student tests and runs programs on the VAX until ready for submission. In this process a sample execution of the program may be captured into a log file which will show all of the interaction. The student then concatenates the Input and Output files, the program listing and any captured sessions into a single "submission file" using the COPY statement. This submission file is then MAILED for marking. As well as each lecturer having a personal account on the VAX, there is a class account for each subject. The student activates a "HANDIN" procedure which automatically transfers the submission file to the class account and an ACKNOWLEDGEMENT OF RECEIPT is sent back to the student.

Students may also MAIL questions directly to the lecturers' personal accounts. In the case of a lecturer being absent from duty, that message will remain unanswered for that period of time. Of course if the lecturer is simply off-campus (at home, at a conference), then by using portable terminals the communication with the students can be maintained regardless of physical situation. I have even used a portable terminal from a telephone box in Melbourne while at a conference in that city, and have been successful in establishing communications even through switchboards with STD bars in larger organizations.

By retaining a separate electronic mail FOLDER for each student in the class accounts, the lecturers have access to the total history of submissions, acknowledgements, questions and answers which have passed between the Institute and a student.

Another feature of electronic mail is the BROADCAST facility which allows a lecturer to send a message to all usernames on a distribution list. This allows for easy and efficient communication of general notes, explanations, and words of encouragement, and because each student must communicate with the system twice a week, the lecturers can assume that such messages are received. The number of LOGINS and duration of connection are stored in the VAX Operating System LOG and the statistics can be distributed to both lecturers and students.

Electronic mail is not restricted to communication between the Institute and the student but also facilitates STUDENT TO STUDENT communication. A number of individuals make regular use of mail to share difficulties and approaches to solutions, and there was a "TUTORIAL" hour established by some students for the exchange of pressing problems.

The use of CAI techniques

With such a brilliant communication facility it was natural that approaches would be made to include elements of CAI into the main stream educational material. Within the Associate Diploma there are already three packages, provided by Digital Equipment Corporation, which are used in a tutorial instruction mode. These are not available to the majority of Post-graduate Diploma students because they require VT100 compatible display terminals and single keystroke responses rather than line oriented responses (this is a costing restriction rather than an AUSTPAC restriction).

The CAI "TUTORIAL" method, while offering an enormous potential for individualized instruction, also places a significant load on staff time in the development phase of the tutorial construction. Various sources have quoted a ratio of 100:1 (100 hours of staff time required to produce 1 hour of tutorial). Again, even though there will be savings in staff time once the tutorial programs are in place, the shortage of staff time for the development cycle renders the tutorial approach practically unviable at present.

The first local CAI developments have focussed on reducing the amount of hack marking by lecturers, and improving feedback of results to students.

Marking programs

The first of these required the students to create a file of answers to multiple choice questions and then to MAIL the file as a submission. Once the submission arrived at the class account, the file was read into a marking program which tallied the score and MAILED the mark, grade and correct answers to problem questions back to the student. These marks were then automatically transferred to a spreadsheet for assessment determination by the lecturer and for other statistical operations (e.g. correlation between student responses).

Computer based feedback

The second program to be developed actually asked the student the questions and accepted the answers interactively. The program allowed the student to move forward or backward through the questions, or tc jump into a particular question. There was also the facility for asking for help with the question where more details of what was required would be given. When the answering of questions was completed, the program would then tell the student the mark which had been achieved (without indicating the questions which were answered incorrectly) and it would list the page numbers of the text and the notes containing the information relevant to the incorrectly answered questions. The student then had the option to submit the current attempt, or to re-read the references and then answer the questions again. The student decided what level of mark (and effort) was acceptable; there was no restriction on the number of attempts by the student. If an attempt was to be submitted then the program would MAIL the answers and the marks to the class account for processing as earlier described, except that in this case the student already knew the mark for the submission.

This second program is used in an Introductory Systems Analysis subject which is based around a fairly large and detailed text. The students find reading the text quite a chore, and often skip or skim read sections. Traditional methods of motivating students to do this reading thoroughly would require the setting of various discussion questions or short answer tests. To be of best use, these in turn would have to be quickly marked and have comments attached, before being returned to the student. There are insufficient staff resources to allow this approach to be effective.

The Computer Based Feedback approach breaks the 100:1 ratio of the tutorial approach (VAX based and PC based programs in use have typically a 10:1 ratio), because most of the required questions being presented in the exercises are available as True/False or Multiple Choice questions to be found in most instructor's manuals. All that remains is the task of locating a page reference which will assist the student should the incorrect answer be given. By allowing multiple attempts at the set of questions, most students do not just accept the grade and go on to the next task, but rather review the references and try again. This can be contrasted with the traditional testing methods where students will often not review the results of a test and, in some cases, not even read the comments appended to their submissions.

In fact, the motivational aspect of using this delivery method became apparent when we included a counter of the number of attempts which individual students were making at the tests. These ranged from 4 or 5 up to 20 to 30. In an attempt to reduce the pressure on terminal lines we distributed printed copies of the questions, so that students could note down answers as they read the text. This decreased the duration of each test (as the student did not have to carefully read the question as presented), but had only minimal impact on the number of attempts.

Exam results

When we compared the final exam results of the post-graduate students with a group of on-campus degree students who also used the same text and sat the same exam (but who did not have access to the Computer Generated Feedback), we found that while the distribution of marks was similar, the post-graduate students' results were centred around a higher mean. In the following semester we made the CBF system available to the next group of on-campus students and experienced a similar upward shift in the mean mark for that group. A dramatic upward shift in mean for a cohort of Associate Diploma in Computing students, who were also allowed to use the CBF system, occurred in the same semester.

The quite dramatic increase in performance, as measured by the examinations (consisting of 100 short answer questions), has also been noticed at QIT following the introduction of Computer Based Training programs by the Educational Research and Development Unit (ERDU), and reported at Calite '86 (1).

Reasons proposed for this success have been:

- that a lot of students no longer study effectively; be this due to lack of knowledge about how to study, or lack of motivation (2)
- that with this method the student is provided with only positive feedback, and references to the correct answers, and so the student does not "learn the distractors" (3)
- that using the computer to answer the questions sustains an interest in the subject material which not only extends the time a student spends on the material, but also makes that time more effective (4).

Conclusion

Whatever the reason, Computer Based Feedback seems to:

1. WORK, in the sense that students achieve a better learning program regardless of their distance from the Institute.

2. WORK, in the sense that students get appropriate feedback at the time of submitting their answers.

3. WORK, in the sense that lecturers are freed from repetitious marking and record keeping, and can devote time to addressing areas of difficulty identified by the incorrect answers to particular questions.

When Computer Based Feedback is combined with a fast response electronic mail system, and is provided over a low cost communication facility, the effects of the physical isolation of the external student are significantly reduced.

References

Ellis, H.D. (1986) Computer based education at Queensland Institute of Technology. *Proceedings of the Fourth Annual CALITE Conference. Adelaide 1986* (pp 92-96).

Ellis, H.D. (May 1987) Private Communication.

Whitelaw, M. L. (July 1987) Private Communication.

Gregor, S. (June 1987) Private Communication.

Don Munro entered the computing industry in 1967. He has worked 11 years in commercial computing environments and 9 years in academic computing environments. He is currently a senior lecturer in computing at the Riverina-Murray Institute of Higher Education, and Course Co-ordinator of the Post-graduate Diploma in Computer Applications. His efforts on Computer Based Feedback programs are a practical response to both the difficulties experienced by the external student, and a desire to see "cost-effective CAI" being incorporated into the mainstream teaching activities of educational institutions. Much of his approach to the design and implementation of CBF programs was investigated during the Autumn semester 1987 when he was granted study leave to visit one commercial and four educational institutions in Australia, and to review their application of computing in their teaching programs.

A case study in computer education by external study

John A Palmer, Ken Peirce, and Graham Pervan
Curtin University of Technology

This paper covers the introduction of a Business Systems unit to the Bachelor of Business degree at Curtin University of Technology and the changes in teaching approaches and techniques which became necessary when it was decided to make the unit available to external students. EDP Systems 200 was developed in 1984 with the objective of providing an introduction to Data Processing and Programming for second year Accounting students. It became a required unit for their studies and was taken as an elective by students elsewhere on the campus.

In 1986 it was decided that the unit should be made available to external students throughout Western Australia because many Accounting students were held up in their studies through not being able to study EDP Systems 200 which was a pre-requisite for other units. This unit was designed in 1986 and implemented for the first time during the Autumn Semester 1987.

This paper describes the design, implementation and problems associated with bringing a unit with many different requirements to a relatively small number of students spread over a vast area.

External Studies at Curtin

From its inception in 1968 the Western Australian Institute of Technology (now Curtin University of Technology) has recognised that it has a responsibility to provide tertiary education to WAIT students regardless of their location, age, status and domestic or work responsibilities. These problems frequently mean that students are unable to complete studies in the normal internal mode.

In order to overcome these inherent problems Curtin offers Undergraduate and Postgraduate programmes for external students in the areas of Business and Administration, Arts, Education, Social Sciences, Engineering and Science, Health Science and Mining. A total of thirteen courses are available entirely by external study and a further twenty seven are partially available. The majority of the students enrolled in these courses are located over widespread areas of Western Australia, and in addition there are significant numbers of students who live in other States and overseas.

External Studies in Business

In addition to a general Graduate Diploma in Business, the Division of Business and Administration offers three of the Majors from its undergraduate Bachelor of Business degree; these are Accounting, Financial Management and Economics, and Marketing. In addition, some Graduate Diploma courses are also offered through external study. All five Schools within the Division are involved to some extent, due to the fact that the degree includes a common core of units from each School. Three of the Schools which also offer a major have greater involvement. The amount of involvement in external studies for any particular unit depends on the degree of difficulty in conducting that unit over large distances.

The School of Computing and Quantitative Studies services two of the common core units and one of the specialist units in the Accounting Major, EDP Systems 200.

EDP Systems 200 at Curtin

Historical background

In 1984, at the last review of the Bachelor of Business Degree, it was agreed that all second year Accounting students should study a unit which would give them some knowledge of both data processing and programming, including hands on experience with microcomputers and an IBM type mainframe computer.

A course was prepared for on-campus students. It was a five hour per week unit made up of studies in the following areas:

- Data Processing
- Introduction to dBASE II
- Introduction to COBOL programming

After two semesters it was decided that COBOL did not satisfy the objectives of the unit and another mainframe programming language was selected. The program chosen was SAS, which was easier for non-specialists of computing to learn on the Amdahl computer because it removed the need for the students to overcome the complexities of Job Control Language (JCL) and concentrate on quickly developing applications. In addition, it provided a sound preparation for the use of the language later in the study of Auditing.

In 1986 dBASE III was introduced to replace the earlier dBASE II. This has been retained for 1987, but will be upgraded to dBASE III Plus for both internal and external students next year.

The unit has proved satisfactory as an introduction to these key areas and many students are using it as a stepping stone into further computer studies. In addition some faculties other than Business have made the unit part of their curriculum. It is now usual for over 400 students to enrol in EDP Systems 200 each year.

The unit requires students to attend lectures in each of the three segments, workshops for the programming sections and tutorials for the data processing component.

The student evaluation consists of continuous assessment for the D.P. area; this is made up of case study work plus tests and an examination. The programming sections are assessed from hands -on assignments carried out in the workshops.

In 1986 it was agreed that EDP Systems 200 should be made available to external students spread over Western Australia, but not open by external study to students in the Metropolitan area of Perth. This decision was made to avoid the overscheduling of available laboratory facilities (already highly utilised) for internal student

Developing the External Unit

The idea of using computers or learning some aspect of computing by external study is not new. The Open University has been doing this since the early seventies (Bramer, 1980). In the case of EDP Systems 200, there were specific problems which appeared different from other attempts. Other hands-on computing units have often involved the study of the BASIC language (for example, Jennings and Atkinson,1982). As EDP Systems 200 involved some IBM mainframe requirements as well as microcomputers and a mix of applied systems software, these past experiences were of little use in developing this unit for external study.

There were three main problems to be dealt with due to this overall decision. The first one was to convert the Data Processing segment from a lecture and tutorial mode to a format compatible to distance learning concepts. The second problem was to develop a microcomputer "hands-on" segment for dBASE III which could be handled by students in remote areas with very little reference to Curtin University. The third goal was to create a teaching programme for SAS on suitable mainframe computers in remote areas.The authors of this paper were given the responsibility of producing this programme by the end of October 1986

Before any of these activities could begin, it was necessary to ascertain the interest and likely enrolment of eligible students and also to find out the possible availability of both microcomputers and IBM type mainframe computers in the country centres. A circular was sent to all prospective students explaining the format of the total unit, advising of the availability of the unit in 1987 and requesting a response regarding the student access to both IBM compatible PCs and mainframes. In addition, teaching institutions in country areas were contacted to see if any of the facilities needed were available at these centres and if so, what conditions would apply to the use of their equipment and teaching staff.

The overall response to these letters was good. There were over thirty replies and from the information given it was apparent that the bulk of the students would be able to gain access to microcomputers. Many had PCs at home, others had them at their work place and other individuals were able to use equipment at local High Schools. This meant that there was no need to use the institutions which had been contacted, although there had been good support from those bodies.

The response regarding the availability of mainframes indicated that it would not be feasible to handle this segment of the course off campus and plans were made to conduct a concentrated course on SAS at Curtin University during the extended Easter break.

Once this information was at hand it was possible to commence preparation of the course material. In the areas of Data Processing and dBASE III, a series of topics were selected and readings were chosen to supplement the topic presentations. A variety of exercises, both assessable and for practice, were included in the topic material, so as to provide a comprehensive set of study notes for the students. For SAS a set of introductory notes were prepared, so as to give the students a basic idea of the language prior to attending the session on campus.

Once enrolments were completed in February 1987, all students were telephoned to determine the type of PC which was accessible and whether all students would be able to attend the on campus session at Curtin for the SAS. Answers to these questions indicated that the students knew the requirements before hand and they all appeared able to conform to the programme. Questions from students were dealt with and they were left with a clear understanding of the objectives and format of the unit.

This initial contact proved very helpful, in that the students commenced their studies with very few problems and misconceptions about the unit. They knew what to do and they already had received personal counselling from their contact in the unit.

The Outcomes of the First Run

Following the unit development as described in section 2.2 above and after the communication with the students and any necessary counselling, it was agreed that the unit was ready for a trial and it was run in the first semester of 1987. The activities involved in each section of the course and the overall outcomes are set out below.

General D.P. Section

The data processing section was generally suitable for changing from internal mode to one satisfactory for use with external students. The general principle was to use the text book (Principles of Data Processing, Ralph J. Stair) as a basis for learning about hardware and software. Readings and other references were used to supplement any deficiencies in the chosen text book. It is usual that one book will rarely cover all the topics in a set curriculum. The first task was to develop individual modules for the unit, each of these to cover a specific section, such as Computer Input. If the text book adequately covered the area the modules were not large, but they were extended if it was considered that the text coverage was insufficient or if the particular topic was not dealt with at all. There were exercises within the text book which the students were expected to complete. These were used for practice only and were not assessable. Assessable exercises were provided for areas not covered by the text book and were also used to test whether the students were able to use the knowledge gained in practical circumstances. An example of this was a case study dealing with an organisation which was having problems with its data processing and was considering introducing a new computer system. Other exercises dealt with areas such as Data Flow Diagrams and Cost/Benefit Analysis which were not covered to any degree in the text.

The final assessment was an end of semester examination which mainly covered material from the relevant parts of the text book.

dBASE III Section

The objectives of the dBASE III section were several. One was to give the students adequate exposure to microcomputers, handling floppy disks, formatting, copying files, backing up data, and appreciating the role of the operating system. Secondly, we wanted to expose the students to concepts of data storage, fields, records and files, and how data in files can be accessed. The vehicle chosen for the achievement of these objectives was dBASE III, although originally the course had been based on dBASE II.

The largest single task in preparing the dBASE III part of the course was writing the external notes for students. Fortunately we had a good basis for starting these notes. For several years we had been teaching dBASE II and dBASE III to the public and providing detailed notes as well as sample data. These notes provided the basis for the external course notes.

A complete set of internal course notes does not suffice for an external course, however, and much of the material had to be rewritten. Unlike teaching in a classroom or even a laboratory, in distance learning the notes are the students' only source of information. It was necessary, therefore, to include in the notes sample screen layouts and detailed explanations which were not part of the original notes. We knew that the first set of notes would be experimental and that a significant re-write would be required for the second offering of the course. As this was a significant part of the teaching in the undergraduate Accounting degree - in fact the students' only exposure to microcomputers - we knew we had to put a lot of continuing effort into the development and refinement of the notes.

Although we were licensed to copy and distribute a tutorial version of dBASE III, we could not provide our external students with any version of MS-DOS. Our internal students were provided with the software legitimately but the external students had to find it themselves. Many of them did not know what an operating system was, let alone how to acquire MS-DOS!

Fortunately the solution to that problem was also part of the solution to the problem of the acquisition of hardware. Once the students had access to an IBM PC or compatible, it was a simple matter to ensure that the student had a copy of DOS. The fun began when we started to get phone calls from students asking whether their particular micro was an IBM compatible. There were machines we had never heard of before - neither had our computer centre.

In time, we had all of the students equipped with PCs or compatibles as well as MS-DOS or PC-DOS. They needed compatible hardware because we were supplying dBASE III and several databases on 5 1/4 inch floppy disks formatted by MS-DOS. These disks did not contain any means with which to boot the system. That had to be done by the students' version of MS-DOS prior to loading our copy of dBASE III. Although we had no responsibility to provide the operating system, we nevertheless took the responsibility of teaching the students how to use the micro, how to boot the system, copy files, rename files, look at the directory, and so on. A simple set of exercises had to be devised to enable the students to gain confidence in manipulating files under MS-DOS. This had not been done for the earlier course in dBASE III offered to the public, so we had to undertake extensive writing of notes.

The final, and somewhat late, aspect of the teaching of dBASE III externally was developing an assignment for assessment. By the end of the dBASE section of the course the students had created their own databases, sequenced them in difference ways, extracted data according to quite complex conditions, and reported on the data extracted. It was not difficult to set an exercise to test a student's ability to create a report on a particular database. This was in fact done by using the CREATE REPORT command. Similarly we could test the student's ability to create a database. The problem arose when we tried to test the student's knowledge of interactive commands, the execution of which we had no way of witnessing. It became necessary to teach the concept of stored commands; in effect we had to teach the students how to create programs. As it turned out, the students coped very well with the concept of command files as well as being able to sequence and extract data from files and store these commands in a program.

SAS Section

The on-campus session used SAS on an Amdahl (a plug compatible IBM mainframe computer) and proved to be successful. At the start of the semester, the students were supplied with notes on SAS and its implementation on the Amdahl in a workpack. A "short answer" test/assignment was include with this material. This simply tested whether the students had read the material. It was required that this work should be submitted prior to the on-campus session. This material gave an introduction to the types of commands, screens and keyboard which the students would be using at Curtin. A concentrated three days made up of lectures (about one day) and laboratory work (about two days) enabled them to learn enough to write a reasonably complex application and then fully test and debug the program. All the students successfully completed this programming assignment and commented favourably on how much had been achieved in such a short time.

The approach taken was similar to that previously used by the authors in the execution of professional short courses for people learning the basics of SAS on an IBM mainframe. The methods used had proved successful in the professional courses and were no less so with these external students. In addition the students expressed appreciation for being given the opportunity for face to face contact with their lecturers which is not usual in the isolation of normal external studies.

Student Performance

On the completion of the semester, 20 students out of the 22 original enrolments had completed all the requirements of the unit and all of these had achieved good results. The average result percentage was 72%, which is better than the average for other units. In addition the retention rate was very satisfactory.

Student response

Once the semester had ended and the results had been distributed, a simple questionnaire was sent to all students to determine their reaction to the unit and also asking for any recommendations in regard to changes to the unit. The areas which were covered in the questionnaire are set out as follows:

- Administration of the unit
- Organisation of the unit
- Academic content:
  - Data processing
  - dBASE III
  - SAS

General comments and recommendations.

Due to the smallness of the group it was not feasible to make valid statistical tests of the answers to the questionnaire, but the overall response indicated that the students did not believe that there was any need for major changes to the current format.

Conclusion

In summary, the first run of the course has gone surprisingly well. No major problems have arisen in any of the three parts of the course and so no major revisions are intended for the next offering in first semester, 1988. Student performance was above expectation and questionnaire responses indicate general satisfaction with the course. As expected, the general data processing component ran smoothly, probably because the requirements were no different from a normal distance learning course, and there was plenty of experience to call upon in that area. The SAS component also, as expected, ran successfully, given the input of previous experience to its development. The greatest uncertainty lay in the success of the dBASE III section, but the care taken (and previous short course experience) led to a successful outcome.

In addition, early planning and contact with intending students well prior to the semester starting meant that there were very few administrative problems. The only changes contemplated for 1988 will be upgrading to dBASE III Plus and reviewing the text books to ensure that those used are up to date.

Bibliography

Ashton Tate, (1984). *dBASE III.* California: Ashton Tate.

Bramer, M., (1980). Using Computers in Distance Education: the First Ten Years of the British Open University. *Computers and Education*, 4, 293-301.

Fillery, P & Palmer, J.A. (1985). *Introduction to Personal Computers.* Perth: Curtin University of Technology.

Jennings, P & Atkinson, R.J. (1982). Learning Computer Programming at a Distance, *Distance Education*, 3(1), 157-169.

Shelly, G & Cashman, T. (1986). *Learning to Use dBASE III.* Boston: Boyd & Fraser.

Stair, R., (1984). *Principles of Data Processing.* Illinois: Irwin.

John Palmer, Ken Peirce and Graham Pervan all teach commercial computing for the School of Computing and Quantitative Studies and have done so since the School was formed in 1975. All the authors have been involved in the development of several public courses offered to the Perth business community in areas related to the selection of microcomputer hardware and software, and programming in various microcomputer-based and mainframe-based software packages.

CAI — A necessity not a choice

Douglas L. Carthew
School of Small Business and Computing
Adelaide College of TAFE

This paper sets out to explain some of the reasons Computer Aided Instruction is not being used by Lecturers to prepare their own material for presentation of self directed learning. The simplicity of text based authoring to a large extent has not been adopted in the classroom. The writer has been involved in the development of a text based system suited to authoring by staff with minimal computer familiarity. The student has control at all times, and provision has been made for branching facilities based on response and time. Emulation of multi-field screens allows for business systems training, and videodisc access is also part of the program facilities. Widely available for evaluation in TAFE the program requires little more than word processing skills for preparation of text. Few lecturers have elected to use the package.

The integrated authoring facility developed by pragmatic response to TAFE lecturers requirements is the "Q" instruction package. This paper addresses the issue of introducing C.A.I. into traditional classrooms where computers are unfamiliar.

Classrooms are designed for presentation of a familiar lecture. C.A.I. caters for the individual learner learning at their own rate, often in their own time.

Professional Educators in TAFE (S.A.) were seeking alternative teaching methods and looking for a C.A.I. package which could be used on already purchased NEC APCIII computers. This required the simple skills of a Wordprocessing Typist and the minimum management skills of lecturing staff.

Aspirations for C.A.I.

Dedicated educators gain reward from student learning. The Richmond College in Victoria had pioneered the application of the package "Author". The package required an IBM with Colour for graphics presentation. Skills required were minimal but suitable hardware had not been purchased by TAFE in Adelaide.

Many lecturers inspired to use Computer Assisted Instruction lacked the determination and zeal to continue application of the most simple software. Graphics C.A.I. was left to the few with the necessary zeal.

Pioneering S.A. TAFE efforts in C.A.I.

The testbed for the emerging methodology will undoubtedly be the new college being developed at Tea Tree Gully, currently under construction.

A pioneering site for the new technology using the "Q" package was Croydon Park. The College was an ideal environment in which to develop the package from a University Module to a System for wide application in TAFE.

The following screen illustrates the system options available within the package.

```
               M A I N   M E N U

Catalogue/EDIT existing TEXT ......
Create new TEXT ..................
CONVERT session TEXT for use by Q .
UN-CONVERT session files to TEXT ..
Test-run Q .......................
Print a copy of CONVERTED TEXT.....
Make an authoring work disk .......
Create/Edit a USER authority list .
........... Q U I T .............
```

Staff Development took place in conjunction with the trials at Croydon Park and "Q" found application in Matriculation studies and Adult Literacy.

Lecturer Hopes

Student Directed Learning

With packages written today using C.A.I., lecturers know the their needs and the students. The packages are available but when the time comes to convert prepared text for presentation on computer, the problems appear greater than the desire to improve methodology.

Curriculum Question Bank

Education administrators are applying pressure to meet rising costs and increase productivity. They are recognising the need to have lecturers more efficient but are not always committed to the application of computer technology and the costs involved in implementing the new methodology.

Administrators know that the curriculum can have assessment on computer. The "Q" package is capable of holding a question bank with all the questions for all the curriculum in TAFE on one large database, and when required, extracting questions for a lecturer from the database. The program exists but directors responsible for implementation need to be willing to give direction and implement the technology. Without the support of lecturers staid in traditional teaching methods administrators are doomed to failure.

Off campus Tutorial

Student learning is leaving the classroom and taking place at home. Adelaide College of TAFE is one of the leaders of distance education in this country.

More and more pressure is being applied by our society, by students who recognise the need for further education, and those seeking advancement in present or future employment.

Many are buying computers to enable them to both study and prepare assignments. They can learn any subject with their computer using C.A.I. at home and be assessed using the same package in the college.

The following Illustrates the system options available to the administrator/manager/lecturer.

```
               M A I N   M E N U

Modify session delivery options ....
Recover data from a user disk ......
Edit/read a QRESULT.TXT file .......
Test-run Q .........................
Print a copy of session text .......
Make a new user (student) disk .....
Create/Edit a USER authority list ..
........... Q U I T ...............
```

Misunderstanding

Most new methodology comes under criticism with a resistance to change. C.A.I. is no exception. Lecturers resist, administration fails to see value, and funding institutions are loath to spend the money.

Administrative Direction

Good managers are often misunderstood by educators who feel their creative teaching efforts are unrewarded. Of all the functions in our society today, education needs sound direction. If senior management are unwilling to support change, then the enthusiastic Lecturer will introduce new methodology "doing their own thing" with unnecessary duplication of effort while looking for leadership.

Coordination needed

Innovation needs direction, and without it, an otherwise successful project may become a failure. Creative staff are difficult to manage and inventors are noted for working outside guidelines which should be imposed. The development of C.A.I. requires a team effort, the support of a director, a willing typist and a facilitator who will participate in the development of the new methodology in the classroom. Encouragement is needed for the student participating in the use of the new methodology.

Work involved

Computer Assisted Instruction requires both dedication to the machines use and a conviction that focus should be on the learner not the lecturer. The Lecturer must be persuaded that C.A.I. will aid in presentation of material in the same way that the overhead projector is used in the classroom or a slide-tape session may be included in the methodology.

Enthusiasm

Students greet Computer Assisted Instruction with a zest born of a fascination with the new technology and the desire to achieve. Using C.A.I. in the classroom daily has convinced me of its value in most subjects. Students pair and answer questions with both active participation and discussion.

Students

All students having overcome fear of technology value simple self directed learning. The difficulty is not the student but the lecturer feeling threatened by computers. The role for the lecturer changes with C.A.I. to one of providing support for the learning environment.

Lecturers

Some lecturers have adopted Computer Assisted Learning, but the majority lose enthusiasm when the full realization of the commitment becomes known. Few realize that they already have much of the basic material prepared in lesson plan form and hand-outs.

Success

Success in alternative teaching methods is symbolized by the use of overhead projectors in the modern classroom.

The recognition of student satisfaction in completion of a "Q" session needs to be experienced. Once experienced by an open minded lecturer, that experience will become motivation for more material preparation.

Australia's software crossroads

A well known phenomenon is the loss of qualified staff to overseas countries. The emergence of significant user friendly Computer Assisted Instruction by Australian authors may lure Australians seeking recognition to countries willing to pay for their expertise.

Package success

The "Q" instruction package developed to meet the needs of TAFE has suffered the relative ignomony reserved for poets in their own land. British Publishing Houses have taken interest in this software produced in Adelaide, and are seeking to market the product in the large U.S. market.

The modern Personal Computer and software packages have matured. Mature Packages which are simple to learn will succeed. "Q" is one such package. It stands well without graphics on a monochrome display, it performs better when enhanced with full colour graphics, accepting colour frames.

"Q" has the ability to interact with other computer packages providing tutoring capability with direct links to another computer program. It is suitable for use in the computer arena while compatibility with Videodisk rates the package with todays leaders providing the very best Video quality graphics with simple Wordstar generated text.

Management solution for friendly software

Like all successful computing installations, Computer Assisted Instruction Management Assessment is simply another one which will stand or fail on the support it receives from senior staff.

Conclusion

Educators today, have may choose to be part of the change, but tomorrow there will not be a choice. Students directing their own learning will demand packages they can take home, packages which they can use to tutor their children, and packages which can be easily modified. Wordprocessing, the tool we use daily, will be the prime tool for preparation of tutorial material. The "Q" Instruction Package succeeds in making minimal assumption of computer familiarity. Its success will be recognised in Australia or overseas. It is to be hoped that the phrase "A man is not without honor but within his own country" will not come to apply to the developers of the "Q" instruction package.

Interactive videodisc in the teaching of Orthopaedics in Phy^siotherapy.

Allan Christie
School of Physiotherapy
South Australian Institute of Technology

The use of an innovative educational medium such as the interactive videodisc was seen as a way of maximizing student learning by using its attributes of excellent visual images, interactivity (active student learning), feedback, quick response time, and the potential for student-directed and student-paced learning. A videodisc containing 1981 slides and approximately 34 minutes of video was produced in PAL format and the authoring language PC/Pilot was used for programming. Evaluation of the effectiveness of interactive videodisc as a learning medium will be conducted using the illuminative method rather than the more traditional outcome-oriented or comparative model.

The teaching of the subject "acute orthopaedics" is performed over two years of the undergraduate physiotherapy course at SAIT. Firstly, there is the expansive but necessary theory input in second year with little practical supplement and, secondly, there is the relatively large practical input in fourth year with little chance to revise the orthopaedic theory except from notes taken in second year.

Student-paced and student-directed learning was seen as a way of addressing these problems, however, a conventional form of self-paced study using student guides/notes was not considered suitable as much of the information to be learnt was of a visual nature. Also feedback and interactivity were considered to be important as there appears to be general acceptance in the literature that learning is an active, rather than a passive, process (Copeland, 1981; Young, 1984; Pemberton et al., 1985; Clark & Sandford, 1986). The interactive videodisc (IVD) system (combining the microcomputer with a videodisc player) was seen as an ideal medium as it presents a flexible image-rich learning environment which has the potential to be both student-paced and student-directed. Second year students could use the system to learn new material and at the same time it would be available to fourth year students to supplement their practical experience.

In accord with Baker & Ziviani (1986) the use of an information transfer system such as IVD was also considered beneficial for the following reasons;

- having subjects available for student observation is desirable though not always possible.
- even when they can be observed, repeated performance for the benefit of student analysis cannot be guaranteed.
- the financial and time costs of organizing such sessions are becoming increasingly prohibitive.
- the instruction is consistently reproducible.

Learning Theories

Learning theories have their basis in two principal schools of psychology - cognitive and behaviourist. Bruner (1966) and Stenhouse (1975) are leaders in the field of cognitive psychology which emphasizes the importance of the process of learning and learning by discovery. Behaviourist psychology was popularized by American psychologists, Tyler (1949) and Skinner (1958). The hallmarks of this approach are behavioural/performance objectives, outcomes of learning and the stimulus-response model. Not surprisingly, the behaviourist approach to learning formed the basis of early programmed instruction methods and subsequent computer assisted learning (CAL) initiatives. Until the mid-1970s much of the evaluation of CAL was also based on this behaviourist approach with the traditional hypothetico-deductive model (in the guise of comparative studies) being used. There was, however, growing dissatisfaction with this approach to evaluation which sought to analyse a complex situation such as the learning milieu in terms of outcome only. A few of the pitfalls of this approach include;

a. not recognizing features salient to the success or acceptability of the project because the information to look for has already been decided at the beginning of the project in a way which excludes "unsought" information.
b. the assumption that if an "experimental" model is adopted that salient variables can be identified and controlled for; an assumption unlikely to be met in a real and complex situation.

Parlett and Hamilton (1977) highlighted these and other shortcomings of the "objectives" approach to evaluation in a book "Beyond the numbers game" and proposed a more comprehensive and descriptive form of evaluation which they called the illuminative method (as light was thrown onto the topic under consideration). Following on from this approach Jones and O'Shea (1981) indicated that the traditional question always posed when considering CAL; "How effective?", should be replaced by "How is the project perceived from the client's point of view - and what are its salient and critical features which make its acceptability and success more or less likely?". In this way one would be looking at both content and quality of the learning experience as well as the context in which learning occurs. Methods used to obtain information using the illuminative approach include;

- observation
- questionnaire (both open and closed questions)
- interview
- written appraisal

Interactive Videodisc

According to Sweeney (1985, p. 90) the following goals should be set and fulfilled when developing a videodisc;

1. utilize all optical videodisc functions in a variety of ways.
2. cohesive topical program even though it contains individual modules.
3. human quality demonstrated as well as archival information storage.
4. high repeatability value - one or two viewings should not exhaust information available in a sequence.
5. should be amenable for setting up programs for learners interested in varied (but related) topics and provide accessibility to different levels of information or expertise.

Interestingly, the author independently developed the Orthopaedic interactive videodisc with similar goals in mind. Put simply, the aim is to use the pedagogical potential of interactive videodisc to its fullest.

IVD Workstation

The IVD workstation used at SAIT consists of a microcomputer, mouse, videodisc player, and monitor. The instructional design of the videodisc material required a single screen presentation so that the broadcast quality video pictures could be overlaid with computer generated text and colour graphics.

Two different single screen configurations have been used by the author;

1. Computer video mixing - video mixing card (Eg., Videologic MIC-2000I + EGA) combined with a fast-blanking monitor (Eg., Sony KX-14CP1).

Pros

- MIC-2000 card initializes the videodisc player and so the same set of commands controls all of the popular videodisc players thus simplifying programming.
- MIC-2000 card can perform soft-editing, i.e., real-time editing of the video, audio and computer signals.
- the single card satisfies both major colour television standards; PAL and NTSC.

Cons

- expensive - MIC-2000I + EGA (Quadram Quad+ rec.) ~A$3400
- recommended monitors such as the Sony KX-14CP1 do not support EGA.

2. Monitor video mixing - e.g., Mitsubishi monitor EUM-1471A combined with EGA card.

Pros

- monitor supports CGA/EGA/PGC.
- PAL video can be overlaid with computer text and graphics via the RGB analogue unit.
- reasonable cost - monitor A$1600 + EGA.

Cons

- programming is more difficult and so there is a time/productivity cost.

The CAL lessons were developed using the PC/Pilot authoring language (see Appendix A).

Economic. Evaluation

Materials

• dubbing high band tape to 1" C format tape and to U-matic low band time code tapes (for off-line editing) (8 hours @ $160/hour)	A$1,280
• use of 1" editing facility (31 hours @ $150/hour)	4,650
• 1" premaster tape	144.60
• slide production	908.64
• high band tapes (17 @ $25 - Sony KCS-20BRK)	425
• broadcast video unit hire (5 days @ $30/day)	150
• royalties for music	80
• single-sided videodisc pressing charges (Sony)	4,840
• 10 videodiscs @ $57.85	578.50
	A$13,056.74

Filming and off-line editing were performed by the audio-visual services unit of the Institute.

Man-hours

• scripting	50 hours
• editing videotape	60 hours
• layout of material on disc	40 hours
• cataloguing slides	60 hours
• programming	200 hours
	(approx) 410 hours

This time (approx. 410 hours) was spread over a period of 12 months but certainly these figures indicate that a videodisc must be produced by a team (ideally consisting of a content expert, instructional designer, programmer and video producer).

Hardware costs continue ɔ decrease but videodiscs and associated software are labour-intensive products, so there is not likely to be a decrease in their cost in proportion to the hardware.

Pedagogical Evaluation

As indicated in the introduction, innovative educational technologies have been inappropriately and inadequately evaluated in the past due to predominance of "scientific" comparative studies. Consequently, the author plans to evaluate this interactive videodisc using, what could be termed a holistic approach, and producing a multifaceted description of its use in the learning milieu.

Evaluation tools will include:

Content	written appraisals
	– physiotherapists
	– orthopaedic surgeons
	pretest/posttest – second year undergraduates
Process	attitudes/opinions
	– questionnaire
	– interview
	observation
	computer program
	– in/out time
	– audit trail through program

Appendix A

An example of the ease of programming using the PC/Pilot authoring language is the following segment of code used to control a videodisc player connected to a microcomputer with a MIC-2000 card and then display a computer generated menu.

```
r:— open MIC device driver
fx:mic
r:— carriage return is used with each
   command to videodisc
r:-- player so it is defined as a string
   for ease of programming
d:cr$(1)
c:cr$=chr(13)
r:————————— videodisc command ——
r:— initialise videodisc player
fo:0,"init"!!cr$
r:————————— videodisc command ——
r:— search for frame 5 and stop videodisc
   player
fo:0,"still  5"!!cr$
r:————————— videodisc command ——
r:— turn video on
fo:0,"video  on"!!cr$
r:————————— videodisc command ——
r:— turn audio on
fo:0,"audio  on"!!cr$
r:————————— videodisc command ——
r:— play videodisc player from frame 5
   to frame 1200
fo:0,"play  ,1200"!!cr$
r:————————— videodisc command ——
r:— wait until end of play
fo:0,"wait  endplay"!!cr$
r:————————— videodisc command ——
r:— turn video off
fo:0,"video  off"!!cr$

r:— main orthopaedic menu 1
j:@p
*menu
ts:m0;e7;v1,38,1,23;e0;f6
t:
:            MAIN MENU 1
:
:    ———————————
ts:f14
t:
:                ORTHOPAEDICS
ts:f4
t:
:          1.  Introduction
:
:          2.  Fractures
:
:          3.  Fixation
:
:          4.  Main Menu 2
:
:          5.  Main Menu 3
:
:          6.  Exit
:
ts:f6
t:    ———————————
:   PRESS #7.>#6 TO SELECT
:   PRESS #7SPACE BAR#6 TO CHOOSE ANOTHER
e:
p:
u:menu
ts:f2;g10,7;a1.    Introduction
*no1  c:x=key(0)
ts(x=13!x=49):g3,21;f7;ALoading......;
l(x=13!x=49):intro
j(x>49&x<54):number
j(x=54):quit
j(x<>32):no1
ts:f4;g10,7;a1.    Introduction
ts:f2;g10,9;a2.    Fractures
*no2  c:x=key(0)
ts(x=13!x=50):g3,21;f7;ALoading......;
l(x=13!x=50):frac
```

```
j(x>48&x<54):number
j(x=54):quit
j(x<>32):no2
ts:f4;g10,9;a2.    Fractures
ts:f2;g10,11;a3.    Fixation
*no3  c:x=key(0)
ts(x=13!x=51):g3,21;f7;ALoading......;
l(x=13!x=51):fixat
j(x>48&x<54):number
j(x=54):quit
j(x<>32):no3
ts:f4;g10,11;a3.    Fixation
ts:f2;g10,13;a4.    Main Menu 2
*no4  c:x=key(0)
ts(x=13!x=52):g3,21;f7;ALoading......;
l(x=13!x=52):menu2
j(x>48&x<54):number
j(x=54):quit
j(x<>32):no4
ts:f4;g10,13;a4.    Main Menu 2
ts:f2;g10,15;a5.    Main Menu 3
*no5  c:x=key(0)
ts(x=13!x=53):g3,21;f7;ALoading......;
l(x=13!x=53):menu3
j(x>48&x<54):number
j(x=54):quit
j(x<>32):no5
ts:f4;g10,15;a5.    Main Menu 3
ts:f2;g10,17;aE.    Exit
*no6  c:x=key(0)
j(x>48&x<54):number
j(x=13!x=54):quit
j(x<>32):no6

*quit
ts(x=13!x=54):f7;b0;m2
ec:
ts:f4;g10,17;aE.    Exit
ts:f2;g10,7;a1.     Introduction
j:no1

*number
ts(x>48&x<54):g3,21;f7;aLoading.....
l(x=49):intro
l(x=50):frac
l(x=51):fixat
l(x=52):menu2
l(x=53):menu3

*no2a
u:menu
ts:f4;g10,7;a1.     Introduction
ts:f2;g10,9;a2.     Fractures
j:no2
*no3a
u:menu
ts:f4;g10,9;a2.     Fractures
ts:f2;g10,11;a3.    Fixation
j:no3
*no4a
u:menu
ts:f4;g10,11;a3.    Fixation
ts:f2;g10,13;a4.    Main Menu 2
j:no4
*no5a
u:menu
ts:f4;g10,13;a4.    Main Menu 2
ts:f2;g10,15;a5.    Main Menu 3
j:no5
*no6a
u:menu
ts:f4;g10,15;a5.    Main Menu 3
ts:f2;g10,17;aE.    Exit
j:no6
```

References

Baker,J. & Ziviani,J. (1986). Interactive videodisc: two bites at the cherry. Proceedings of 4th Annual Computer *Assisted Learning in Tertiary Education Conference*, Adelaide, pp.38-45.

Bruner,J.S. (1966). *Toward a theory of instruction*. Cambridge: Harvard University Press.

Clark,D.R. & Sandford,N. (1986). Semantic descriptors and maps of meaning for videodisc images. *Programmed Learning & Educational Technology*, 23(1), 84-90.

Copeland,P. (1981). The educational significance of electronic media. In F.Percival & H.Ellington (Eds.), *Aspects of educational technology*, vol XV. London, Kogan Page, pp 231-237.

Jones,A.C. & O'Shea,T. (1981). Evaluation methods in distance computer-assisted learning. In F.Percival & H.Ellington (Eds.), *Aspects of educational technology*, vol XV. London: Kogan Page, (pp 176-182).

Parlett,M. & Hamilton,D. (1977). Evaluation as illumination: a new approach to the study of innovatory programmes. In D.Hamilton, D.Jenkins, C.King, B.MacDonald & M.Parlett (Eds.), *Beyond the numbers game*. London, Macmillan, pp 6-22.

Pemberton,P., Taylor,J. & Toleman,M. (1985). Interactive videodisc and student control of learning. *Proceedings of the 3rd Annual Computer Assisted Learning in Tertiary Education Conference*, Melbourne, 69-78.

Skinner,B.F. (1958). Teaching machines. *Science*, 128, 269-277.

Stenhouse,L. (1975). An introduction to curriculum research and development. London: Heinemann.

Sweeney,M.A. (1985). The nurse's guide to computers. New York: Macmillan.

Tyler,R.W. (1949). Basic principles of curriculum and instruction. Chicago: University of Chicago Press.

Young,J.I. (1984). Videodisc simulation: tomorrow's technology today. *Computers in the Schools*, 1(2), 49-57.

Allan Christie is a Lecturer in Orthopaedics in the School of Physiotherapy, South Australian Institute of Technology. Using the PC/Pilot authoring language, he has developed several computer packages which are used by undergraduate students. He has also produced an interactive videodisc to assist student learning in the field of orthopaedics. Current research interests relate to the use of the interactive videodisc and videotape systems and how to best evaluate these educational technologies in the learning milieu.

Using computers in education intelligently:
What happens when computers are removed from the classroom?

Kelvin W. Duncan
Department of Zoology
University of Canterbury, New Zealand

This is a case study of a first-year university course in which computers were used extensively in the teaching and administration of the course until, following a change in staff a few years ago, the computing component was removed from most of the course work. This paper describes what happened following this change to more traditional ways of teaching.

The intelligent use of computers in classrooms is possible provided the following desiderata are observed: They should be used only when appropriate. They should be cost-effective, time-effective and pedagogically effective. The teaching programs should be comprehensible, relevant and non-trivial with a good student-machine interface. Costly establishment overheads should be minimized. Future hardware incompatibilities should be avoided.

There is a general recognition that the computer is here to stay in tertiary education and that it can make a great contribution to teaching effectiveness and attractiveness. It can facilitate interaction with the subject matter, generating a real involvement and deeper understanding. But to a teacher considering introducing CAL techniques the difficulties in establishing a CAL programme are very large, as are the costs. Teaching with computers requires special skills, and the time required to develop and trial CAL modules can be very great. Costs, for equipment software and assistance, can be so high that they may pose another large hurdle to the development of CAL courseware.

These costs in time and money are recognized well enough and are a reason often advanced as a reason for not using CAL techniques in the class room. But what is not generally appreciated is that rejecting CAL techniques also involves a cost. While academics are accustomed to change in their research fields, it is surprisingly difficult to convince them that a static approach to teaching is not a no-cost option. Lost opportunities for change in search of excellence and more effective teaching can be damaging and thus much more costly in the longer term than if CAL programmes had been instituted.

Rather than consider the difficulties of establishing computer-based teaching methods in a course, I wish to examine the rather novel question of what happens if the new technology is abandoned and traditional techniques reverted to as a matter of deliberate policy. Consideration of the effects of this may give insights into such questions as the following: Is it always intelligent not to use the new technology? Should we always rely on tried- and-true traditional methods? Is there a cost if we do take the conservative approach? Is use of only a part of the new technology worse than not using it at all? In short, is it intelligent not to use the new technology?

It is often not appreciated that the nature of tertiary teaching has changed fundamentally in recent times, so that courses which rely solely on traditional teaching techniques can fall into the following traps even though they may have been successful in the past:

a. The course content (subject matter) may appear old-fashioned and therefore non-relevant. This is especially so if no modern techniques or aids are employed at all. To a generation of students brought up in an environment at home and at school or university in which the new technology - in the form of television, audio, computers, arcade games and the like - makes up a vital part of daily life, chalk and talk can seem very dull. Those courses which reject the new technology and employ only traditional 'spray and pray' teaching techniques may loose elective enrollments with serious consequences for funding if this is based on enrollments.

b. The teachers in courses relying on traditional teaching techniques may appear 'old fashioned' in their approach and out of touch, leading to an impression of mediocrity in their discipline.

c. Course revisions and updates can be more difficult to do. On the other hand, computer-based courseware is easily edited and updated so the days of tatty notes resulting from 'cut-and-paste' editing can be left behind.

d. The teachers using traditional methods in a course may find it difficult to take advantage of new developments or teaching aids because they are locked in to a conservative form of teaching. Over the years, they might find that the eventual expenditure of time and effort in attempts to improve and update their courses may be far greater than if CAL courseware had been developed in the first place.

f. Demands on teaching assistants (markers, tutors and demonstrators) may become too costly in traditional courses. This is especially important now that expenditure on tertiary education is being closely scrutinized.

g. Skill development and learning of content may be less, often much less, than that attainable in a CAL course.

h. In courses taught in a traditional manner there may develop a strong tendency to non–participatory, passive learning leading to a tendency toward surface, or shallow learning. On the other hand, CAL can encourage participatory learning with the development of deep understanding. Furthermore, learning in such courses can be more thorough, faster, more enjoyable, and with more opportunities for remedial instruction than in courses taught in traditional ways.

Let us now examine the fate of one course which fell into these pitfalls because of a reluctance of the teaching staff to become involved with computers and computing.

BIOL10X - A subject which went away from CAL

In the early 1970's elementary biology teaching at Canterbury University, Christchurch, New Zealand, was rationalized and reorganized into more logical units. Thus BIOL10X, amongst other biology courses, was begun. It was to present material which had been taught reasonably successfully earlier in a different course format. However, the reorganization caused the staff to think about their effectiveness, and so new computer-based methods were tried.

Early on in the history of the course various computer simulations were developed. These were extended and improved as experience accumulated. Later, the learning was modularised, and computer-based tests and practice programs introduced.

The students were allowed to progress more-or-less at their own pace. When they felt competent, they sat a computerized test of the material covered in that module. The computer program devised the questions from an approved pool devised by the teachers; it marked the answers, kept the student records and made recommendations to each student regarding remedial work for areas of weakness. Using a mastery model of teaching, failing students had opportunity to re-sit until they achieved a sufficient standard to advance to the next module.

Thus at the end of 1978 the course made heavy use of computers for the following purposes:

1. Simulation
   a. of biological phenomena impossible to demonstrate otherwise;
   b. of practical work done by students. The computer simulations acted as checks on the students' progress and comprehension of students;

2. Analysis
   of the results of student experiments using statistical methods.

3. Practice
   Practice programs and quizzes were available for non-compulsory review.

4. Testing
   Using marksense cards, a system of quizzes, which terminated and tested a sequence of instructional modules, was instituted. At the end of each unit of work the student sat the test. Retests were available. Marking and comprehensive analysis was done by computer; the machine producing suggestions for remedial work for each student if necessary.

5. Administration
   Intermarker performance was monitored and compensated for if necessary.
   All the ordinary administrative tasks were performed by the computers.

This was all before micros made it much easier to carry out such work - these applications were done in the 1970's on mainframes, and were largely carried out in batch mode to minimize cost.

The teachers of BIOL10X were enthusiastic and worked well as a team. New approaches were encouraged and no compromise was made to dilute the subject matter. Difficult topics were attempted, and usually conquered. The students were encouraged to go beyond the confines of their course work and to work on their own to greater breadth and depth.

The course was very popular as was shown by replies to independent questionnaires administered by the student body. These surveys indicated that most students found the course to be very worthwhile and interesting, with a well-structured and logical theory and practical component.

The popularity of the course was also indicated by the greater proportion of the class taking the course on a voluntary basis than in any other biology course. Enrollments in some other biology courses were nearly as high, or even higher, but their numbers were boosted by prerequisites making enrollments compulsory. The high number of elective enrollments in BIOL10X were a matter of pride to the teaching staff.

A change of scene

Departmental policy was for staff to be rotated around courses at intervals of some years to "keep them from going stale". In due course the staff of BIOL10X were replaced, in spite of their considerable opposition and reluctance.

The new staff were far less keen on computers. A few simulations were retained, but much less emphasis was given to the heavy involvement of computing in the course work. So it proved impossible to maintain 'mastery' type learning where each individual maintained his/her own rate of progress, and their attainment and progress was closely monitored and remedial work set in place immediately.

The long-termed consequences for this course were disastrous. From a vibrant, well-regarded course it became voted "the worse course at the University of Canterbury" within a few years of the change over. This had a terrible effect on staff morale. Now the course is notable for the lack of cooperation between the staff, the lack of staff enthusiasm and involvement. Students are bored and merely do the course because they are forced to because of prerequisite demands. Elective enrollments have disappeared.

Table 1, which gives the class sizes in BIOL10X since establishment, shows dramatically the decline in numbers of elective students in the years following removal of computers. There are, of course, other explanations for this decline; of which three spring to mind: (1) the subject matter may have become less popular or unfashionable; (2) there may have been changes in the structure of university enrollments such that the size of the elective pool became progressively smaller; or (3) the new team of teachers were not as good as the old.

There is no evidence for any change in the popularity of the course. Very similar courses at other universities have shown steady growth - not the dramatic decline experienced at Canterbury. It is significant that CAL plays an ever-increasing role in the similar courses at other universities.

Nor is there any solid evidence for the second explanation - if anything, students can take more elective courses now than in the late 1970's since many prerequisites have been relaxed.

The third proposition, that the new teachers were not as good as the original team, is debatable since the previous teachers were, in the opinions of their colleagues, rather bad! One was notorious for reading his lectures from a disjointed series of texts - he never wrote lecture notes and merely presented a series of quotations which were often not in a very logical order. The students, too, were often critical in their questionnaire replies, of the lecturing style of the original teachers, even when they sang the praises of the course as a whole.

The most probable explanation is that the removal of computers caused a variety of things to happen which resulted in the decline in student enrolment. Firstly, much of the old teaching programme had to be discarded or drastically simplified to fit the constraints of a lecture-based course. This resulted in boredom and a lack of challenge amongst both students and teachers. Learning became passive and non-participatory which increased student dissatisfaction. The previous close monitoring of student progress and provision of feedback had to be given up, which resulted in increased student anxiety about their progress and a focussing on marks rather than mastery of the subject. The students no longer felt as though they could conquer the course incrementally, but had to master it all in one step for finals examination. This, too, increased student disquiet. And they no longer felt that the course had a discovery learning component. Instead, they had to work at a pedestrian pace in large classes with minimal staff help or enthusiasm on material which seemed dull and devoid of challenge. The staff also felt the changes. From a closely planned course it became a set of individual teaching efforts, uncoordinated to the whole. The 'my patch' syndrome developed, and a lack of cooperation affected every teacher.

Further evidence as to what went wrong, here are a selection of quotations from the course reviews conducted by the students and based on the replies to questionnaires.

1978 (the first year of these independent course assessments) "An interesting course. Well structured and logical practicals and lectures."

1980 "Depressing. Course made up of dregs of all other Bio. courses. Poorly structured. Labs confusing. Pathetic."

1982 "This course should be avoided at all costs. It is incredibly boring. Labs badly organized, dull and irrelevant. Dedication is required to complete. Only bright spot is that assessment load is light, that is if you can fight back the boredom long enough to do some work."

1984 "This is the worst course at the University of Canterbury. Avoid at all costs."

The course enrolment is protected from becoming zero by enforced enrolment on account of the prerequisite requirements of advanced second and third year courses. If a student wishes to advance to these second and third year courses he must take BIOL10X. We can estimate this enforced enrolment by adding the number of students in advanced courses with BIOL10X as a prerequisite plus first year failures and drop-outs. The figures for this are given in the second last column of Table 1. By calculating the percentage of elective enrollments (given in the last column of Table 1) it can be seen the voluntary enrolment in the course fell rapidly as the course became unpopular until today there is no evidence for any elective (voluntary) enrolment at all.

The change had even more serious consequences for other biology courses. BIOL10X was a flagship course where experimental methods of instruction and teaching were welcome and enthusiastically trialled. After a successful trial in this course many very valuable innovations were adopted in other courses. Following the changes, the flagship function disappeared. Finally, the enthusiasm of the old, and now disbanded team for CAL techniques was dissipated. They had been taken against their will away from a course in which they had close personal involvement, and placed in new courses with new colleagues indifferent to, or even positively hostile to, computer-based teaching. Somewhat naturally, the old enthusiasts ran out of steam. Faced with new, hostile environments it proved difficult to maintain enthusiasm for the new style of teaching with its great establishment overheads. As a consequence, we now lag behind in the CAL area, not merely in the particular subject matter taught in BIOL10X, but in most other aspects of biology as well.

It is trite to say that computers are here to stay and that they have a vital and neces-

TABLE 1. Class sizes in BIOL10X in different years. At the end of the 1978 academic year computers were removed from the classrooms. The column headed enforced enrolment gives the estimated number of students who had to take the course for prerequisite reasons.

Year	No of Students	Enforced Enrolment	Non-Compulsory Enrolment (%)
1975	245	98	60.0
1976	252	101	59.9
1977	260	103	60.4
1978	248	100	59.6
(Computers were removed at the end of 1978)			
1979	270	111	58.8
1980	232	113	51.3
1981	227	126	44.5
1982	218	115	47.2
1983	176	109	38.1
1984	179	110	38.5
1985	210	135	35.7
1986	125	125	0.0
1987	139	139*	0.0*

* Estimated

sary role to play in teaching. But the converse of this is not trite, and is often not recognized: to not use computers may seem like the easy option, but it is not a "no cost" one.

Kelvin Duncan is Senior Lecturer in Zoology, he has been using computers in teaching since 1967. He has worked with a variety of UK and USA groups interested in CAL, including Computers in the Undergraduate Curriculum, University of London, University of Lancaster, and others. Main interest area lies in the use of computers in simulation and analysis in biology practical classes, but he has also worked with primary and mentally handicapped children. Beside major professional research work on ecological physiology of terrestrial invertebrates, he has been working on ways of making the computer more accessible to intellectually handicapped children in the belief that computers have a major role to play in their education. To this end he works with a number of teacher groups on a voluntary basis to develop courseware for schc ls.

Software Simulation of Microprocessor Hardware

David T Edwards and Robert G Craig ,
School of Engineering
Darling Downs Institute of Advanced Education

This paper reports on a TurboPascal language simulation program running on personal computers being developed to aid the teaching of microprocessor fundamentals to both internal and external undergraduate Electrical Engineering students of the Darling Downs Institute of Advanced Education. Traditional teaching has used a Motorola MC6802 based single board development system (the D3 or D5), where program debugging is achieved by halting the program and sequentially examining the CPU registers. The enhanced display capabilities of this simulation package allow students to simultaneously view memory and register contents as a machine language program executes. Preliminary trials with internal students indicate a significant speed up factor in the learning process.

The D.D.I.A.E. School of Engineering has for a number of years been teaching microprocessor fundamentals to students enrolled externally in the Associate Diploma in Electrical Engineering. To develop hands on skills, these students have attended an on campus residential schoo.. They have then carried out a programming assignment using a microprocessor development "kit in a suitcase" - the D3 kit based on the Motorola MC6802 microprocessor. The kit was collected at residential school and "passed around" among a group of students to complete their assignment(s). Internal associate diploma and degree students use a similar microprocessor development system, the D5 board, which is a later but near identical version of the D3 board.

One of the authors (Edwards) has developed a software simulator for this Motorola D3 (and D5) kit. This program, written in the TurboPascal language, is intended for running on an IBM-PC or equivalent. Access to such desk top computers is made available to students in the D.D.I.A.E. external study centres located throughout Queensland.

Compared to the original hardware the simulator has a number of significant enhancements designed to expedite the teaching/learning process. An early version of the simulator was used on a trial basis in first semester 1987 with an internal associate diploma class. In second semester 1987 the simulator is being used with an external associate diploma group and an internal degree group.

This paper will briefly describe the existing hardware system. The simulator software and the results of the first semester trials will then be discussed.

Objectives of developing the 6802 Simulator

The principal objectives of introducing the 6802 simulator were twofold.

1. To provide the same learning opportunities for external students as exist for internal students.

As internal students have class access to the hardware for approximately 22 hours, and a large amount of "out-of-hours" access, spread over the whole semester, external students should have a similar amount of "hands-on" access. The present "pass-the-kit" situation for external students limits their hardware access to about 10 hours "driving instructions" at residential school plus one intense burst when they have a kit at home. To provide each external student with a kit of their own is prohibitively expensive at a component cost of approximately $1000 per kit plus many hours of technician time both to build up the kits into a self-contained entity and to service them when faults occur.

2. To improve the laboratory support for the teaching of microprocessor fundamentals.

Students find difficulty in coming to grips with many of the essentials of microprocessor programming. Many find concepts such as flags, program counter, stack pointer and indexing quite difficult to grasp. Displaying these and other features simultaneously should enhance the students' learning.

The existing hardware has no conveniently accessible off-line program storage so students must either key in their programs each time by hand or down-load from one of the Institute's main-frame computers (an option not available to external students). Easy access to off-line program storage is a needed feature.

Description of existing hardware

The D3 kit is a combination of hardware and firmware designed to assist with the understanding of the Motorola MC6802 microprocessor and commonly associated components. The system as configured by the School of Engineering has the following facilities :-

- A ROM monitor which performs house-keeping and provides the user with a number of utility routines.
- 256 bytes of RAM for user program and data storage.
- A keypad for inputing either hexadecimal code or invoking some of the monitor routines.
- An array of 7-segment digits to display both user and monitor information.
- A tape input/output port for magnetic storage.
- An RS232 interface for serial communications (ACIA).
- A programmable interface for parallel communications (PIA).
- A reset button which clears the CPU registers and returns control to the system monitor.

The above configuration permits the students to progress from the design of simple data manipulation programs to the design and implementation of more complex real-time algorithms such as digital filters and interrupt driven feedback control. Unbuffered access is available to the system address, control and data buses enabling such operations as DMA, or the accessing of the digital output of an A/D converter directly onto the system bus, as alternative input/ output mechanisms.

Specification of the D3 simulator

1. The simulator should have the same display as the actual D3 kit.

This is a 6 digit 7-segment hexadecimal display with the first 4 digits for display of the address of a memory location and the last two digits for display of the contents of that memory location. (see figure 1).

2. The keyboard input to the simulator should mimic the keyboard input to the D3 kit. The D3 keypad has 25 keys arranged in a 5 by 5 matrix (see figure 2).

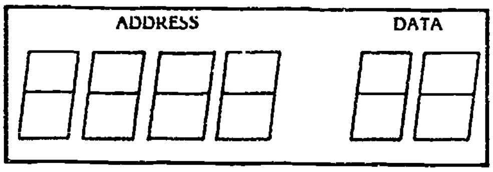


Figure 1: Data display

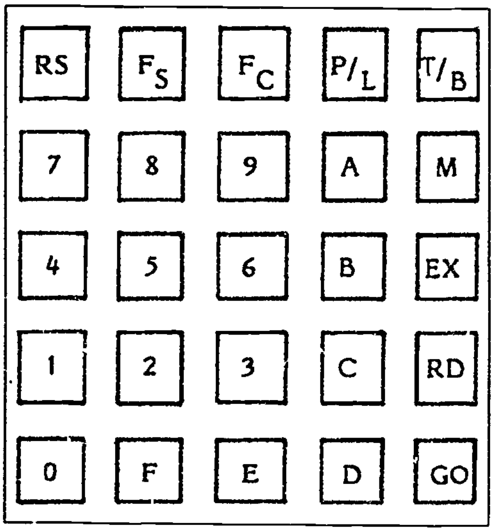


Figure 2: D3 Keypad

The 16 keys for inputting hexadecimal digits (0...9,A...F), for address or memory contents should be implemented by the corresponding keys on the simulator microcomputer keyboard. The remaining 9 keys perform housekeeping by accessing the Motorola monitor. Since these keys have a multi-character label, a suitable single keyboard character would have to be found to be used on the simulator. The D3 function keys have the following actions:

Key	*Function*
RS	reset the CPU registers and invoke the monitor 'prompt' routine
FS	select alternative key function
FC	clear alternative key function
P/L	tape store/load
T/B	step a machine language program one instruction; or insert a breakpoint
M	allow editing of tne memory location the address of which has been entered; or allow access to the previous memory location when editing
EX	invoke 'prompt' without resetting the CPU registers; or cease the execution of a running program
RD	CPU register display
GO	run a program; or allow access to next memory location when editing

3. All needed Motorola 6802 opcodes should be simulated.

(The only opcodes not implemented are those that refer to interrupts :- RTI, WAI, SWI.)

4. Sufficient of the Motorola D3 monitor routines as is necessary for students to complete their assignments should be implemented.

Many of the monitor routines are effectively implemented through the keyboard keys. In addition the following monitor routines were required:

GET	waits for a key to be pressed and then loads that key's "value" into accumulator A.
DYSCOD	takes hexadecimal values stored in a buffer (HEXBUF) and produces a 7-segment display code for each digit in the buffer (DISBUF).
PUT	displays on the output display the 7-segment displays stored in (DISBUF).
PROMPT	returns to the prompt and allows further monitor instructions. (The last action at the end of a machine language program).

5. Compared to the D3 hardware the simulator should have enhanced display characteristics. The standard display of numeric information should be in hexadecimal. The display on the simulator screen should include, in addition to the standard display of the hardware:

- The contents of all registers. The contents of the condition code (flag) register should be displayed both in hexadecimal and in binary. The binary display makes it easier to see which of the 6 flags have been set.
- The current position of the program counter (PC) and the stack pointer (SP).
- The contents of the page (256 bytes) of user memory. This display should distinguish between values that have been set (or defined) by the user as program or data and the "random" values that the memory locations have either from previous use or switch-on. (This option was chosen to unclutter the display.)
- Any changes to SP, PC, register or memory contents caused by an instruction being executed should be highlighted.

6. There should be a capability to load machine language programs (memory contents) from disk and also to save to disk.

7. On-line help should be available.

For external students all the information required to use the simulator should be available on the single floppy disk.

8. All functions of the simulator should be "menu-driven" with single key stroke choices at each stage.

The simulator program should be "crash-proof" with the user protected against making mistakes that could cause problems.

Description of the simulator program

The program has been developed using the TurboPascal language. This was used for a number of reasons including:

- The friendly development environment.
- TurboPascal compilers are readily available for a number of different desktop computers.

This means the source code can be ported from one m[illegible] to another using a file transfer program like "Kermit" and then recompiled on the new target machine. (The original simulator program was developed on an NEC-APC 1 running CP/M-86 and then ported to the MS-DOS IBM-PC compatibles).

The Electrical Engineering laboratories at the D.D.I.A.E. have a mixture of NEC-APC 1's and CCS IBM-PC compatibles. The external study centres have a mixture of Sperry, Hitachi and CCS IBM-PC compatible computers. For the wide range of target computers, there was a need to make the system as machine independent as possible. All screen displays were implemented using only standard ASCII characters for compatibility reasons. The only "graphics" capability used is screen highlighting to improve the readability of the display.

The program consists of almost 3000 lines of Pascal source code divided into almost 100 procedures and functions. Half of the procedures were required to implement the 6802 opcodes. An "Include" file is used to handle the different program requirements of the CP/M and MS-DOS machines. (This only occurs to handle differing screen highlighting requirements imposed by different "IBM-PC" graphics screen/board combinations.)

An "Include" file is also used to handle the different requirements of the D3 kit as used by external students and the D5 kit used by internal students. The differences are :- the page of memory available ($00 on D3, $E3 on D5) and the addresses of the monitor routines and buffers.

To meet the design criteria the following principles were adopted:

1. A portion of the lower left-hand corner of the simulator display mimics the 6 digit display window of the D3 kit.

2. Most of the 25 keys of the D3 kit keypad have been implemented.

Simulator key	*D3 key*	*Function*
0...9,A...F	"same"	hexadecimal digits
R	RS	reset
T	T/B	single step
M	M	memory
G	GO	go
X	EX	return to prompt
V	F/S - T/B	set/clear breakpoint

As well for the simulator keyboard there is a "hidden" menu of simulator only keys.

L	load	allows a machine language program to be loaded from disk
H	help	displays help screens
S	save	saves a machine language program to disk
Z	zero	clears the simulator display
Q	quit	quit the simulator
P	peek	allows the contents of a single memory location (or register) to be displayed in binary as well as the standard hexadecimal
O	observe	displays the contents of the more commonly used monitor buffers
Y		display a trace of previously executed machine language instructions (up to 20 since last reset)

3. Help is available in two forms.

a. On-line Help screens to explain specific features of the simulator.

b. Provision of a number of sample programs (exercises) together with an explanation of their operation.

All help files are plain text files that can be modified using any standard editor without the need to recompile the simulator program.

4. Display enhancements.

Screen highlighting is used to make more readable and obvious any changes to the display caused by the execution of an instruction. Because different hardware/screen combinations give different effects when highlighting is used, the user is offered a choice of three different highlighting options:- higher intensity, inverse video, or none.

5. Loading and Saving.

When loading or saving a file the user is offered a choice of file names from a menu. This limited choice of file names avoids the problem of users forgetting their file names or choosing file names that clash with simulator system files. Enhancements of the simulator over the "real thing" include ability to save a program to disk, on-line help available and much more informative display capabilities (see figure 2).

- memory contents of a full page
- simultaneous display of all registers
- disassembly of the last instruction executed
- dynamic display of program counter and stack pointer positions
- highlighting of changes caused by last instruction
- ability to examine memory contents in binary rather than hexadecimal
- ability to see trace of last 20 instructions
- distinction on screen between defined and un-defined memory contents
- simultaneous display of position of all breakpoints

Limitations of the simulator.

1. Only one page (256 bytes) of user memory available. The D3 kit as used by external students has only the single zero page available. A multipage D5 version of the simulator was tested but was felt to be confusing with the need to switch between pages of display as the areas of memory being accessed changed.

2. I/O is not simulated. Programs cannot interact with the real world.

3. Interrupts are not simulated.

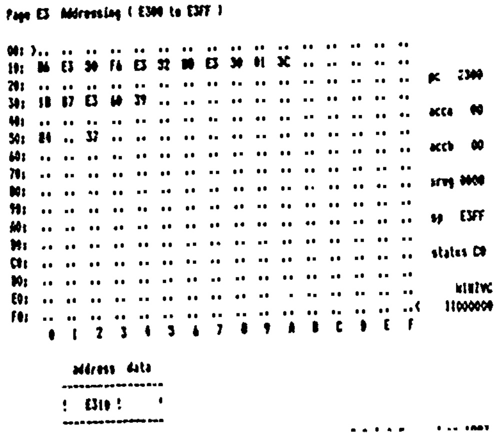
4. Only a limited number of the monitor routines are simulated.

5. On start up the program counter and stack pointer are initialised to the top and bottom of memory respectively rather than being unspecified.

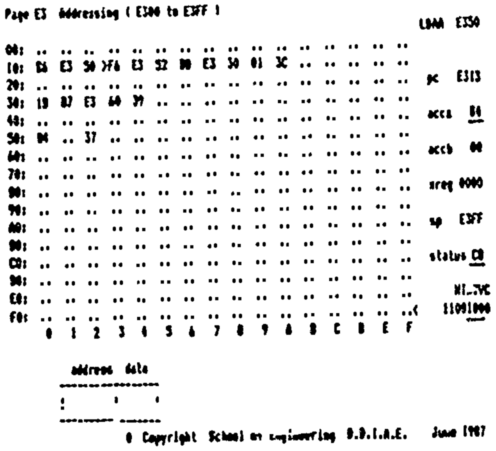
Operation of the simulator

Figures 3 to 7 show the computer screen as a stored machine language program is run.

Figure 3 (below) shows the screen with the program loaded from disk and the starting address ($E310) keyed in.

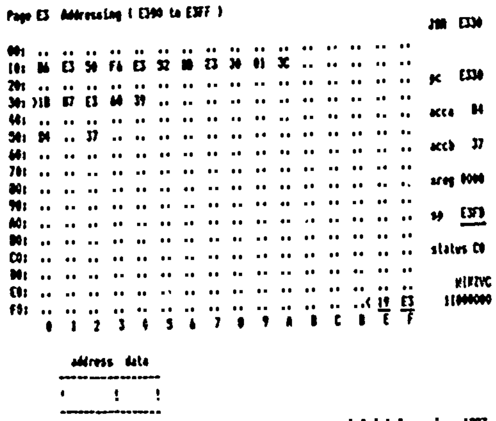


```
Page E3  Addressing ( E300 to E3FF )

00: )..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
10:  B6  E3  50  F6  E3  52  BD  E3  30  01  3C  ..  ..  ..  ..  ..     pc   E300
20:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
30:  1B  B7  E3  60  39  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     acca   00
40:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
50:  84  ..  32  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     accb   00
60:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
70:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     sreg 0000
80:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
90:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     sp   E3FF
A0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
B0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     status C0
C0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
D0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..        HINZVC
E0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..       11000000
F0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..<
      0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F

        address  data
        -----------------
        !  E310 !       !
        -----------------
```



```
Page E3  Addressing ( E300 to E3FF )
                                                                        LDAA  E350
00:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
10:  B6  E3  50 >F6  E3  52  BD  E3  30  01  3C  ..  ..  ..  ..  ..     pc   E313
20:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
30:  1B  B7  E3  60  39  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     acca   84
40:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
50:  84  ..  37  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     accb   00
60:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
70:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     sreg 0000
80:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
90:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     sp   E3FF
A0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
B0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     status C8
C0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
D0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..        HINZVC
E0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..       11001000
F0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..<
      0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F

        address  data
        -----------------
        !       !       !
        -----------------
                 © Copyright  School of Engineering  D.D.I.A.E.   June 1987
```

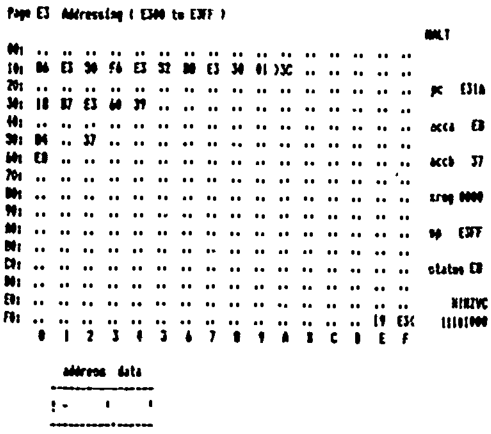
Figure 4 shows the screen after the first instruction has been carried out. The disassembled instruction (LDAA) displayed in top right-hand corner and screen changes shown highlighted (underlined in figure).



```
Page E3  Addressing ( E300 to E3FF )
                                                                        JSR   E330
00:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
10:  B6  E3  50  F6  E3  52  BD  E3  30  01  3C  ..  ..  ..  ..  ..     pc   E330
20:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
30: >1B  B7  E3  60  39  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     acca   84
40:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
50:  84  ..  37  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     accb   37
60:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
70:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     sreg 0000
80:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
90:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     sp   E3FD
A0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
B0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..     status C0
C0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
D0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..        HINZVC
E0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..       11000000
F0:  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..<19  E3
      0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F

        address  data
        -----------------
        !       !       !
        -----------------
```

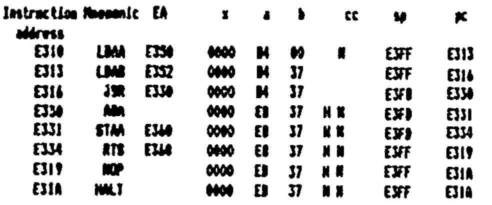
Figure 5 shows screen after jump to subroutine (JSR) with stack loaded (bottom right-hand corner).

Figure 6 (on the next page) shows screen at end of program (HALT used to end simulator programs).



```
Page E3  Addressing ( E300 to E3FF )
                                                              HALT
00:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
10:  B6 E3 50 F6 E3 52 BD E3 30 01 3E .. .. .. .. ..
20:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..     pc    E31A
30:  1B B7 E3 60 39 .. .. .. .. .. .. .. .. .. .. ..
40:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..     acca  EB
50:  B4 .. 37 .. .. .. .. .. .. .. .. .. .. .. .. ..
60:  EB .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..     accb  37
70:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
80:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..     xreg 0000
90:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
A0:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..     sp    E3FF
B0:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
C0:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..     status E8
D0:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
E0:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..        HINZVC
F0:  .. .. .. .. .. .. .. .. .. .. .. .. .. .. 19 E3     11101000
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

        address  data
       ----------------
       ! -      !     !
       ----------------
```

Figure 7 shows the trace option with all program steps being displayed.



```
Instruction Mnemonic  EA       x     a    b     cc      sp      pc
 address
   E310      LDAA    E350    0000   B4   00      N     E3FF    E313
   E313      LDAB    E352    0000   B4   37            E3FF    E316
   E316       JSR    E330    0000   B4   37            E3FD    E330
   E330       ABA            0000   EB   37    H N     E3FD    E331
   E331      STAA    E360    0000   EB   37    H N     E3FD    E334
   E334       RTS    E360    0000   EB   37    H N     E3FF    E319
   E319       NOP            0000   EB   37    H N     E3FF    E31A
   E31A      HALT            0000   EB   37    H N     E3FF    E31A
```

Results of student trials

The simulator was trialed during first semester 1987 with students taking the unit 77201 - Engineering Computers. This unit, taught by one of the authors (Craig), introduces students to computer number systems; computer architecture; machine and assembly language programming; input/output methods and interfacing techniques.

In 1987 the students received their introduction to the 6802 microprocessor on the simulator. Sole instruction on the simulator was given for about 6 weeks and then the students were introduced to the actual D5 kit so they could begin using the serial and parallel I/O ports.

Comparison of the progress of this group of students with students from previous years who did not use the simulator, led to the conclusion that the simulator users had picked up the basics of microprocessor operation in about half the usual time. Comments from students who had previously studied this or the similar degree level unit were usually that the enhanced display capabilities were of significant assistance. Small group interviews of randomly selected students elicited a number of possible improvements for the simulator. These included:

1. Providing a selection of simple (sample) programs that demonstrate the usage of various microprocessor instructions (especially the monitor read and display routines). This has now been incorporated in the latest version.
2. Tying in a 6802 cross-assembler with the capability to download to the D3/D5 kit. This assembler is now available for use by semester 2 students.

Conclusion

The simulator differs from those normally found in microprocessor development systems in that it was primarily designed as a teaching aid and not as a software development tool. The consequence of this design philosophy is the great emphasis that has been placed on such dynamic features as the stack pointer, condition code and most importantly the program counter (PC). Being able to watch the PC move through a program (with the registers and memory contents changing as appropriate) has been of great benefit; a conclusion supported by the very favourable reaction from the student users.

Acknowledgements

The authors would like to acknowledge the assistance of fellow engineering staff members who provided valuable input in the initial stages of the simulator design.

David Edwards is currently Lecturer in School of Engineeri'g DDIAE. Teaching interests include classical control engineering and structured programming. Interested in the development of teaching aids for students (both internal and off-campus) using video and print media as well as CML and CAL.

Robert Craig, has been a Lecturer in School of Engineering DDIAE since 1985. Previously involved with military electronics in UK. Main fields of interest are digital design with CAE and microprocessor based equipment for the mentally and physically handicapped. Also interested in the development of CAL packages for distance education.

Innovative Applications of Computer-Assisted Learning.

Margaret Farrow
School of Occupational Therapy
South Australian Institute of Technology

This paper challenges a number of controversial epistemological issues inherent in the common model of Computer-Assisted Learning (CAL) in tertiary education in which individuals or small groups of students sit at a terminal and learn from material which presented to them, that is, the computer (in essence) being used in the traditional 'tutor' role. The central concern is that this model 'dramatizes the absurdity of continuing to think of computers, or about education itself in traditional ways' (Papert, 1979). Two innovative applications of CAL which address these theoretical considerations of education will be outlined.

The development of a framework based on an unsophisticated model which is being used at earlier levels of education enables the use of the computer as a 'tutee', and as a means of aiding and stimulating tertiary students' thinking. The subsequent application of the 'tutee' framework to teacher education is proposed as a means of encouraging teachers to develop an interest in using the computer as tutee, rather than just as a tutor and so become a more useful educational tool. The two applications, developed through co-operation between two educators in different vocational disciplines demonstrates the value of the CAL educator's ability to apply imagination and innovation to instructional design.

Replication of traditional instructional models rather than innovative variations of these techniques has been suggested (Papert, 1979; Sims, 1986) as a reason behind the failure of computers to achieve their potential as an instructional resource. Sims (1986, p. 262) states that 'The versatility of the computer has been neglected through it's use as a surrogate instructor'. Rightly, he argues the need for the training of teachers in the use of computers as instructional tools. Courseware quality, he describes as being dependent upon teachers acquiring knowledge, experience and competency in authoring skills and that 'armed with these skills, the author has the opportunity to be creative in design by using the instructional versatility of the technology' (Sims, 1986). He specifically mentions the need for imagination in utilizing the interactive capabilities of the learning medium. Interactive qualities are frequently discarded in the application of the computer as a tutor, whether the 'mode' be Tutorial, Drill and Practice, Simulation or Testing techniques.

It seems to me that the dilemma faced in striving to achieve the goal of high quality courseware comes largely from the notion that the starting point for the development of all courseware requires the application of either an authoring language or an authoring system. The starting point, I contend, is the nature of the knowledge area in which the educator identifies a need to facilitate or enhance students' learning ability . A statement of the educational objectives underlying the need to learn the particular subject material is essential as a means of defining the nature of the information or knowledge. It is necessary to go back as far as defining the very beginning educational concepts, such as, why it is necessary for students to acquire this information. Having locked ourselves into the notion that authoring languages and systems are the cornerstone for computer-assisted learning it appears that we may be on a misguided mission as we attempt to find subject materials which we can apply to these authoring tools.

As an educator concerned about factors which affect the incentives, efficiency and effectiveness of stud its' learning, I feel a responsibility to recognize the areas of learning which students find particularly difficult due either to monotony, complexity or unfamiliarity with the material or concepts. Having recognized the educational problem, my responsibility extends to exploration of the most appropriate means to alleviate the difficulties. In the production of the interactive video lessons required to provide exposure to real client examples for Health Science students, the importance of an authoring language (PC Pilot) was indeed recognized (Farrow, 1986). In this instance, the identified educational need was for students to transfer static information in the form of facts, diagrams and illustrations into a dynamic, motion based concept as a means of integrating principles, and facts into clinical practice.

Recently, a need has been identified to increase motivation within Health Science students to learn medical vocabulary and the relevant abbreviations which commonly appear in client files, medical reports and student lecture notes. The educational problem is that the content is boring and requires rote learning, not unlike learning to spell in primary school. Although some aspects of the terms do not refer specifically to medical or scientific terminology (Young & Austin, 1979, p. 8) as they are a part of words in ordinary speech, unfortunately, until these terms are learned in a medical context as well as learning the additional specific medical terms, the student is unable to even begin to understand terms used in subjects such as basic anatomy, physiology and the aetiology of medical conditions. There exists an abundance of texts which contain the facts to be learned, the educational need was simply to find a motivating and interesting way of practising the use of the information and not one of collecting and collating together the information to be learned (Steen, 1971; Young & Austin, 1979).

When considering the most appropriate CAL mode for this educational problem, it was felt that whilst the preferred choice, a drill and practice programme would provide students with the opportunity to practice learning and using medical terminology, three major problems occurred in the application of this learning mode. Firstly, the number of medical terms and their abbreviations are in the region of thousands and vary both nationally and internationally. Secondly, there are multiple examples for the prefixes and suffixes used to make up medical terminology and coupled with both ordinary speech words and medical words, the inclusion of extensive examples would necessitate the linking of unrealistically large data files to a drill and practice lesson. To give examples to these problems:-

1. With respect to medical abbreviations; in different contexts, 'LE' may mean left eye, lower extremity or lupus erythematosus.
2. With respect to medical terminology the prefix 'ab' means away from and is used in the words abductor (leading away from), aboral (away from mouth) and many more.
3. A drill and practice lesson is usually undertaken individually by students, an aspect which as an educator, I believe CAL has the potential to overuse with detrimental effects.

An exciting program, Shrink 'n' Stretch (WESOFT,1983) is designed on the 'tutee' (Taylor, 1980, p. 4) in that the computer presents a contraction such as 'wasn't', to be matched by the expanded form, in this case 'was not'. The student is then asked to test the computer on a contraction known by the student. Through this form of scoring game, in which the student scores if the computer either cannot match the contraction presented by the student, or the computer's presentation is matched correctly, the student is contributing, initiating (Hancock, 1985, p. 21) and drawing on knowledge already learned rather than simply following instructions. The program has extensive interactive aspects, an element frequently discarded from otherwise excellent courseware (Sims, 1986). It appeared that, if used by small groups of students, there existed in this type of program, the potential for each student to 'learn something about how his or her own thinking works' (Taylor, 1980, p. 4) compared with other students at the same academic level. Having observed the enthusiasm with which numerous young adults (the age applicable to the majority of today's tertiary students) not only enjoy arcade games but will work apparently tirelessly to beat their previous scores, an attempt to design an adult version of a 'Skrink 'n' Stretch' seemed worthwhile.

Discussion about the feasibility of writing the proposed program led to an arrangement whereby the writing of a 'framework' for the program was undertaken by an educator in the field of teacher education. The basis for this arrangement was that the 'framework' of the program could then be available for trainee teachers to use to develop confidence in using computers in their regular teaching (Anderson 1984, p. 88). By teachers developing and evaluating in the classroom programs based on the 'tutee' model for topics such as capital cities matched to states or countries there is a potential for teachers to become clearer as to why they would introduce the computer to the classroom as an educational tool (Anderson, 1985).

Written in basic for the Apple GS, the program is short and simple and links to a data file into which the educator inserts a limited number of clear or pertinent examples (between 20 a d 100) of subject content. As the student(s) add more known examples the data file increases similarly. It could be argued that the 'framework' is in itself an authoring language, however, the innovative element of the program is the capacity for the student to teach the computer, thus eliminating the need for the computer to hold an exhaustive store of correct facts which must be 'judged' against student responses.

The program has no facility to detect an incorrect answer which has not been included as one of the initial pertinent examples in the data file except by the educator regularly checking the data file as it builds up with further student examples. To add an exhaustive data file of correct answers against which students could check their own answers would defeat the innovative and exciting aspect of the program and once again place the computer in the tutor mode.

Most CAL materials produced are based on the five commonly recognized modes and whether developed using authoring languages or authoring systems continue to perpetuate the tendency for the CAL author to place the computer in the 'tutor' role, the traditional model of the teacher in the educational process. Imagination and innovation have been applied to the instructional design of a program designed to overcome the motivational difficulties experienced by tertiary students in learning basic medical terminology and the relevant abbreviations. Through it's development in co-operation with an educator involved in training teachers and the subsequent application of the program 'framework' as a means to encourage teachers to explore potential uses of computers in the classroom, the applications of this tutee 'framework' challenges a number of controversial epistemological issues inherent in the common CALITE model.

References

Anderson, J. (1984) *Computing in Australian schools: An Australian perspective.* Australian Education Review No. 21. Hawthorn, Victoria: Australian Council for Educational Research.

Anderson, J. (1985) *Computers in the language classroom*. Perth: Australian Reading Association.

Farrow, M. (1986) Interactive video as a specific training medium for health science students. In Bishop, G. & van Lint, W., *Proceedings of the Fourth Annual Computer Assisted Learning Conference in Tertiary Education Conference,* Adelaide: ASCILITE.

Hancock, J. (1985) Computers for Learning. In J. Anderson (ed.) *Computers in the language classroom.* Perth: Australian Reading Association.

Papert, S. (1979) Computers and learning. In Dertouzos, M. and Moses, J. (Eds.) *The computer age: A twenty year view.* Massachusetts: MIT Press.

Steen, E. (1971) *Dictionary of abbreviations in medicine and the related sciences.* London: Bailliere, Tindall and Cassell.

Sims, R., (1986) On the Development of high quality courseware. In Bishop, G. & van Lint, W., *Proceedings of the fourth annual computer assisted learning conference in tertiary education conference,* Adelaide: ASCILITE.

WESOFT (1983) *Skrink 'n' Stretch in punctuation.* Perth: Schools Computer Software, Education Department of Western Australia.

Young, C & Austin, M. (1979) *Learning medical terminology, step by* step. St Louis: Mosby Co.

Margaret Farrow has been a lecturer in the School of Occupational Therapy, S.A. Institute of Technology for past 9 years in areas of physical dysfunction and therapeutic applications. In 1985, developed an interest in CAL through a research project aimed at increasing both self instructional learning opportunities and exposure to actual clinical problems (via videotape) for Occupational Therapy students. With authoring assistance from Rod Sims, she has produced an Interactive video lesson (using PC Pilot) which was trialled with students in 1986, the findings being presented at CALITE the same year. Currently a member of the S.A. Institute of technology's Computer-Based Education Subcommittee of Academic Computing Board and is working towards the establishment of a CAL Unit. Continuing to upgrade and produce Interactive Video lessons for Health Science students at the Institute. In early November, 1987 participating in the ADCIS Conference in Oakland, California, U.S.A.

Computer managed learning in Physiology laboratory classes

R.E. Kemm
Physiology Department,
University of Melbourne.

The Physiology Department has been investigating the enhancement of its undergraduate laboratory teaching by the use of computer based apparatus over a number of years (Delbridge & Kemm, 1985) and has now drawn up specifications of its requirements for such a 'Laboratory Machine' following extensive testing of hardware and software. This paper discusses the results of these investigations using peripheral interfaces connected to the Apple IIe and IBM-PC computers as well as the final specifications for the 'Laboratory Machine' for which funding is now being sought.

The overall requirement for the machine is versatility, so that it can act as an intelligent instrument (incorporating chart recorder and oscilloscope functions) which can measure a wide range of physiological parameters, as a *analyzer* of the data gathered, as a *simulator* of experiments which are too difficult to mount in an undergraduate laboratory, and for computer assisted learning functions to relate and test the students knowledge of lecture material. A fully-integrated computer managed learning package is required to allow the students to swap between any of these modes of operation, without requiring significant computer experience.

Enhanced student teaching in undergraduate laboratories is the aim of replacement of obsolete practical class equipment by computer-based apparatus, referred to as the 'Laboratory Machine'. The 'Laboratory Machine' would permit a more extensive range of scientific laboratory skills to be acquired than with more expensive modern conventional physiological apparatus. The greater versatility of computer-based equipment would allow new experiments to be performed with a more stimulating variety of investigations of physiological phenomena in a standard class arrangement.

This paper is intended to discuss the real problems in the upgrading of the laboratory equipment used in the teaching of undergraduate students in a Physiology department. We are faced with the common problem in many Universities of upgrading our laboratory equipment for undergraduate students in the face of a declining funding and decreasing purchasing power for imported apparatus. Costing of the replacement of our 20-year old obsolete and potentially unsafe equipment by conventional apparatus puts it well beyond our means, even if we phase in the purchases over many years.

We were led to investigate the use of computer based laboratory apparatus by the discovery of inexpensive digital oscilloscopes based on the Apple IIe microcomputer, which we could see as the central data gathering device for most of our laboratory experiments. The installation of such equipment has been carefully considered in a staged programme over a number of years. We thus purchased an Apple IIe Computerscope for evaluation and and then ran a pilot project to investigate its use in a typical class experiment using four such instruments (Delbridge &

Kemm,1985). We were able to compare costs of conventional and computer based apparatus, putting us in a good position to argue their relative merits when seeking funding and also to justify the development and evaluation of a prototype which would allow the development of specifications for the final 'Laboratory Machine'. The testing phase using the IBM PC as a reference machine is virtually complete. It is felt that it would be most cost effective to have a single company tender for the delivery of a whole system to our specifications, so that all components of the 'Laboratory Machine' are integrated into a coherent package and so that a commercially viable completed package is available.

The Department of Physiology and the Medical Faculty of the University of Melbourne have provided approximately $50,000 over the last four years for software and hardware. Two academic and one technical staff member have also had a significant involvement in the project in this period. IBM have also been of considerable assistance by providing up to four computers for development and evaluation.

Specifications of the laboratory machine

We have drawn up a detailed specification of the minimum performance required of the 'Laboratory Machine', although only the general specifications are included here. It describes the requirements of the transducers, amplifiers and couplers, electrical stimulators, data sampling rates and precision, data logging and review, display requirements, data storage and data review facilities. Some of these features are covered in Appendix A.

Although the IBM PC was used as a prototype, the computer to be used is not specified since changes in such hardware are very rapid and the final choice will depend on compromises between cost and facilities offered as well as the ability of the machine to operate a significant range of commercially available simulation software and other packages. Similarly, the choice of a network and file serving computer to distribute software and data should also be delayed. Connected to the computer, we prefer a free standing unit which houses all the electronic couplers and amplifiers, so that the computer can select the appropriate devices required in particular experiments, without the time consuming task of the technical staff having to setup the apparatus for particular experiments. This is especially important when the apparatus will be in much more intensive use than many separate pieces of conventional apparatus.

Overall specifications

The 'Laboratory Machine' comprises a computer with its operating and applications software, together with amplifiers and electronic coupling and the devices which measure or control the physiological parameters being examined. The 'Laboratory Machine' is required to operated in the widest range of physiological studies, using an integrated software package to use the full facilities of the machine.

This computer managed learning package would have four major areas of operation in the laboratory class. These are:

1. As an *instrument* for data acquisition and control in physiological experiments, operating as an oscilloscope, chart recorder and stimulator. The chart recorder function would include a rolling screen display and a slow data-logging mode direct to an ink-jet printer. Data would be directly stored to disk for later review or hardcopy. So that this one instrument can replace a variety of standard physiological apparatus, it must be versatile and easy to setup and use in a wide range of labora-

tory situations. A range of electronic couplers and transducers would be required. The computer should be able to control the sensitivity of the couplers or amplifiers.

2. *Analysis of experimental data* using statistical tests, curve fitting and comparisons with class or published data. In many experiments data gathering would proceed slowly in the background over a long period, allowing concurrent operation of a progressive review and manipulation of previously acquired data

3. *Computer-assisted learning* as an aid to directing students towards various experimental investigations and to relating material presented in the lecture course.

4. *Simulation of experiments* where the procedures are too difficult or expensive for an undergraduate class. This is seen as an adjunct to real laboratory experience and not a replacement. The computer-managed learning package must be able to swap easily between the four areas, with data gathered during the experiment able to be examined or operated upon in any other area.

Operation of this management package must be easy for lecturers and tutors, who will be required to link various routines within a computer-assisted lesson. These lessons would include gathering of data from experiments or simulations, student analysis of data, and questioning the student before and after the experiments. These academics can be expected to be familiar with using computers with standard software packages, but should not be expected to have any professional programming skills.

Management software should be based on a Window and Menuing system, preferably mouse-driven, to give optimum user interaction. Changes in parameters for printing, display, data acquisition rates, stimulus parameters are to be be easily made using on screen displays. A colour monitor is required, particularly for displays of multiple graphs and in simulations involving complex diagrams. Data acquired from experiment or simulation may require further analysis, which is not provided in the package supplied. The data should be able to be exported to standard "user-friendly" packages.

Development of the specifications of the laboratory machine

There have been two major phases in the development of our detailed specifications. These were the pilot project involving the Apple IIe in a typical physiology class and then using the IBM PC as a testbed for new software and hardware which would be used in the wide range of physiological measurements.

Pilot project using the Apple]le

Aims

The specific questions we sought to answer in the pilot project were: Could the computer-based apparatus perform as well or better than conventional apparatus as a data acquisition instrument? Was the apparatus acceptable and easy to operate by undergraduate students? Could computer managed learning improve student understanding of the experiments and their relation to lecture material? Were simulated experiments a useful supplement?

Results

We found reasonable answers to the above questions and indications of future directions which might be appropriate for fully developing computer based apparatus.

As an data acquisition instrument, the Apple Computerscope performed better than an affordable digital oscilloscope. The provision of a hard-copy of the results during experiments was the highlight of its operation for many students.

The majority of the students were confident in the operation of the instrument after a brief familiarisation lesson on the computer, although some students still had an aversion for computers. (Its operation was simplified from the commercially available package by using a specially encoded keypad which allowed access to a limited range of the functions available to an experienced user of oscilloscopes. The software was also configured so that it started up in the appropriate mode for particular experiments.)

The students did show a better appreciation of the experiments they were performing and their relationship to lecture material, although the it was not possible to distinguish between dependence of this improvement on the computer managed learning component or greater interest in the project produced by its novelty.

The simulated experiments were both popular and useful, since they bridged the gap between the lecture material which was more closely related to the data analysed in the simulation and the less direct measurements on the phenomena which could be investigated with class equipment.

Conclusions from pilot project

The major criticism of the apparatus used in the pilot project was that the software available on the Apple IIe for the individual components of computer-managed learning exercise (data acquisition, computer-assisted learning and simulation) could not be integrated on the Apple IIe because of incompatible languages, unsuitabilty of software for user customisation and lack of local support from suppliers. It was also difficult to modify the software to present simple and relevant screen labeling for student use.

The pilot project only looked at one type of experiment which is normally performed in undergraduate physiology classes. There are a wide range of parameters which are normally measured in animal and human experimental physiology and these are indicated by the descriptions of the current experiments (Appendix B) and by the range of transducers and couplers which would be required in the future (Appendix A).

It was concluded that the Apple IIe was not a suitable machine for a major re-investment because of declining support for data acquisition modules and because better software and hardware was becoming available for the IBM PC and Apple Macintosh. It was decided to proceed with the investigation on the IBM PC because of its open-architecture, colour facilities and the interest in local software groups in developing both the data acquisition and computer aided-learning facilities.

A prototype system based on the IBM PC was developed to demonstrate the remaining facility which was not available (a 4-channel chart recorder) and to test the transducers and couplers which would be required for a full range of class experiments.

Development and performance of a prototype laboratory machine (based on the IBM PC)

The prototype was designed to answer two broad questions:

1. Could we source the hardware and software, other than the computer, within Australia?
2. Could the prototype machine provide

satisfactory performance so that we could confidently demonstrate to funding authorities that there was a better alternative to conventional laboratory apparatus.

The specific tests we had to perform to satisfy these requirements were:

1. to evaluate the existing hardware and software which would permit data acquisition and the development of computer aided learning of the type we had investigated on the Apple IIe and to see whether there would be any problems in integrating their operation.
2. to seek and test suitable amplifiers and couplers from local manufacturers.
3. to have a local software group develop 4-channel chart recorder software.

Performance of prototype and comparisons with commercially available apparatus (for student use)

There are many authoring systems and many data acquisition systems available, but there is no integrated approach which combines oscilloscope and chart recorder functions with computer assisted learning and simulations all within the one machine. A number of Physiology Departments in the U.K. and elsewhere have published reports on their use of microcomputers as either data gathering machines in undergraduate laboratory classes or simulations or for computer-aided testing instruction. Many of the small computers (e.g. BBC and Apple IIe) in use are not suitable for a fully integrated approach.

The current state of development of the hardware and software which might be suitable for use in the 'Laboratory Machine' and other sources are described in the following section, together with comments about a few commercial data acquisition systems. It should be stressed that we require an inexpensive system, so that many of the research-oriented devices are unsuitable on price as well as complexity of operation.

Software performance

Laboratory Software Associates (LSA of Melbourne) have produced a pre-release package which shows the feasibility of a 4-channel vertical chart-recorder on an IBM-PC. We tested this extensively in parallel with a conventional Grass 4-channel recorder in a research laboratory which measures many of the parameters likely to be encountered in undergraduate classes. A number of suggestions have been made to improve the package and LSA have provided an updated version for evaluation. A slow chart-record using an ink jet printer needs to be developed.

Their general purpose data acquisition package (DAOS) already has most of the facilities required for the oscilloscope package. This package was originally developed for the DEC PDP-11 computers and received worldwide acceptance. DAOS is now available on the IBM PC, although some work would be required to provide the facilities in the specifications of the 'Laboratory Machine'. The cost of the full DAOS package is too high for general student use ($3,100 per copy with up to 30% quantity discount) Many of its features would not be required for teaching, so a less expensive run-time system would be used on student machines and the full version for program development.

Microcraft (Melbourne) have produced and authoring system ('Author') which would be satisfactory for computer assisted learning. Preliminary discussions between Microcraft and LSA have established the feasibility of linking their software packages. 'Author' also has the ability to be linked to simulation software and would probably form the foundation for the overall computer–managed learning package. The price of this package (approx. $600 for quantity) is reasonable.

There are a number of simulation packages available for a variety of PC's, some of which would need to be re-written for use on the computer chosen.

The major limitation in software is the *overall* management package which is not currently being considered by any of the software groups which have been consulted. It would not be a very difficult task to adapt the Authoring and data acquisition packages to operate within such a management package.

We have examined many alternative systems including MacLab System (Otago, N.Z.) based on the Apple Macintosh This has some features not yet available in the LSA software for the IBM. The latest information we have shows that they have an excellent software package available for data-gathering with oscilloscope functions, and a chart recorder function has been released. There is no computer assisted learning option.

Coulbourn Instruments (U.S.A.) VideoGraph software operates their standard physiological couplers via an IBM-PC, but this appears not to have all the features required for our measurements and has no computer assisted learning facilities.

There are many other packages for data acquisition, but their suitability would depend on price and whether or not they could be integrated into a management package.

Hardware performance

J-RAK Instruments (Melbourne) produce conventional couplers for physiological measurements using standard chart recorders and these have been purchased for evaluation with the chart-recording software on the IBM-PC. So far we have tested the bridge couplers and universal couplers successfully. A prototype Cardiac Output device has been briefly tested and will need some modification before it is used in conjunction with the computer. They also have a stimulator under development.

This department has designed and built, for the Apple-IIe pilot project, general-purpose pre-amplifiers for recording extracellular potentials in man and animal. These have been designed with optically isolated inputs so they could satisfy safety standards for recording from patients. J-RAK have built a similar unit based on this design, which we will evaluate.

When using equipment in large undergraduate classes, it was found unacceptable to have ambiguity in the control of modules by either the computer or by a switch on a panel of a coupler. The conflict between controls on the front panel and any computer controlled gain would have to be resolved by covering the front panels or by modifying the J-RAK modules.

The Otago (N.Z.) couplers for the Macintosh consists only of 4 general purpose electrophysiological amplifiers, with none of the other couplers we require being mentioned as under development. Their amplifiers all have complete computer control of gain. They are working on a stimulator module. Their apparatus currently performs some of the laboratory functions specified. It could be enhanced to satisfy our requirements, perhaps with the assistance of J-RAK, although colour options on the Macintosh II are very expensive and the small screen monochrome Macintosh limits its appeal for complex diagrams.

Coulbourn Instruments (U.S.A) have an incomplete range of physiological couplers and transducers available, but again they are principally designed for operation independent of the computer and would require modification to provide an integrated operation with the computer in control.

Nucleus (Sydney) produce a satisfactory IBM-PC computer interface board to connect to the couplers and this is supported by the LSA software. All other interface boards (which allow timing and conversion of the incoming signals to digital values for the computer) are imported, at greater cost.

There are many special purpose data-loggers and oscilloscope systems based on the IBM-PC, but most are expensive and designed for research purposes. No commercial systems are yet available with a satisfactory range of couplers which are integrated with the computer system.

Comparisons of costs: Conventional apparatus versus laboratory machine

One of the strong arguments for the use of computer based equipment is the potential cost savings. The individual items of equipment are less expensive, and the apparatus is more versatile than conventional equipment, so we will need fewer pieces of apparatus. The versatility of the apparatus means that it will be in more constant use in laboratory classes, and can also be used for student revision outside normal class times.

Our Physiology Department requires 42 'Laboratory Machines' for its normal commitment for undergraduate laboratory classes for 180 Medical, 50 Dental and 200 Science students. These machines could also satisfy the requirements of the Pharmacology department classes, given timetabling adjustments or a slight increase in numbers of machines. Students would work in pairs in most experiments and in fours when the experimental protocol required group investigations.

We examined another University Physiology Department's inventory of modern, but conventional, laboratory apparatus and costed its replacement at current prices. The apparatus included 4-channel chart recorders, 2-channel storage oscilloscopes, stimulators, spirometers, amplifiers, transducers and couplers. The estimated 1987 replacement cost would be over $27,000 per student workplace (assuming work in pairs in most experiments and in fours for the remaining experiments). This compares with our current estimates of approximately $13,000 for computer based equipment. The price of computer based equipment is likely to fall, or its performance increase, while the conventional equipment price could rise further. Also the major cost of the computer system, couplers and amplifiers, would be significantly reduced by larger orders.

Implementing the replacement of existing apparatus — Future directions and financial considerations

Further progress with the 'Laboratory Machine' is now dependent on a major financial commitment by the Physiology Department or others to purchase a commercial number of machines. We are currently seeking funds with some urgency, as our obsolete equipment is becoming unserviceable. Whether the final computer is based on an IBM, Macintosh or other computer, no system has reached the specifications required to perform the wide range of physiological measurements which would be undertaken (Appendix B), let alone the integrated approach of combining data gathering, analysis, simulations and computer aided learning within one computer package.

The individual components required have been demonstrated as being viable, but their final development requires that there be a large enough order to finance the relatively simple task of providing the management software and production runs of couplers. This will require that a department takes the first step by itself, or a group of Physiology and Biological Science Departments form a consortium of buyers to draw up common specifications and call for tenders. The latter approach has many advantages, particularly in the pooling of resources in the development of applications software for simulations and for developing experiments which use the special expertise of each department.

We believe that the 'Laboratory Machine' can be produced for much less than a similarly equipped 4-channel pen recorder and a digital storage oscilloscope, but with all the advantages of computer based apparatus. There seems to be a very large market in teaching and research for the 'Machine' described here and the consortium could attract government interest in the potential for export.

Reference

Delbridge,L & Kemm,R.E. (1985). Computer managed learning in an undergraduate laboratory. In J.A.Bowden & S.Lichtenstein (Eds.) *Student Control of Learning: Computers in Tertiary Education*, Melbourne: CSHE, University of Melbourne, pp. 95-97.

Appendix A

Indicates the requirements for transducers and couplers in the Department of Physiology.

#	Measured parameter	Transducer	Signal Coupler
1	Arterial B.P. FSD 300 mm Hg 0-50 Hz	Strain gauge 2-8 mV	Bridge & D.C.amp
1	Venous B.P. FSD 40 mm Hg (± 20 mm Hg) 0-50 Hz	Strain gauge	Bridge & D.C.amp 1 mV
1	Muscle Force (a) FSD 1Kg (b) FSD 10g	Strain gauge	Bridge & D.C.amp
1	Muscle Length FSD 1 cm	Optical ?	Special Coupler
1	Core Temp. 35-39 °C 0-0.1 Hz	Thermistor	Bridge & D.C.amp
1	Resp. Flow & Volumes 0-40 Hz	Turbine etc	Special Coupler
2	Extracellular Potentials (FSD's) ECG .03-150Hz, 10mV EEG .05-50Hz, 1mV EMG 10-6KHz, 10mV NERVE .03Hz-10KHz,10mV	Specific Ag Electrodes	Optical-Isolated Pre-amplifier
1	PCG 5Hz-2KHz	Microphone	A.C. Amplifier
1	Cardiac Output (Imp. Method)	Monitoring Electrodes	Special Interface
2	Auxiliary Inputs (general use) Frog Skin 200mV	Devices with own bridge etc. with Offset,	Differential D C. Amp. Amplification, Attenuation & Filtering.
1	Double Pulse Stimulator	------	Stimulus Isolator

Appendix B

Outline of requirements of current laboratory experiments and the 'laboratory machine' in first and second year Physiology classes.

The variety and technology required to perform experimental physiology is not widely appreciated, so an outline of the experiments currently performed is given below. As well as describing current experiments for 2nd Year physiology classes, indications are given of the easy upgrading of these experiments. However, new experiments will also need to be devised, or adapted from other institutions, to further modernize the teaching programme. Final year experimental physiology experiments are not included, since they are usually performed in small groups and are more easily adapted to change in equipment or emphasis by the lecturing staff. We would expect to be able to present students with challenges much closer to the experimental physiology they might encounter in their future workplace.

One of the significant advantages of the new equipment is that more experiments will be able to be performed on human subjects with the greater data gathering and analytical power of the computer based apparatus.

Experiment 1: Permeability of Cell Membranes

Demonstrates the factors influencing the transport of materials across membranes using erythrocyte and yeast cell.It requires no computerized data acquisition or any particular laboratory

Computer Simulations could considerably assist the students understanding of diffusion, osmosis, isotonicity and active transport mechanisms. Computer assisted learning would also be important for this conceptually difficult area of physiology.

Experiment 2: Characteristics of Skeletal muscle

Examines the factors which influence the development of force following electrical stimulation of the isolated toad muscle. Fast data acquisition, in oscilloscope mode, of muscle force and length is required, with the possible addition of recording the electrical activity of muscle (EMG) and receptors. New transducers and muscle baths are required, particularly if another muscle (e.g.sartorious) is used to allow direct stimulation of the muscle, rather than via the fatigue prone neuromuscular junction used in sciatic/gastrocnemius. It is possible to introduce human muscle studies using EMG and force measurements (if additional heavy duty force transducers are purchased). This may be an optional facility for students proceeding to exercise physiology.

Experiment 3: Action potentials in peripheral nerves

The current experiment examines the characteristics of Action Potentials initiated by electrical stimulation in the sciatic nerve of the toad. This requires fast data acquisition in oscilloscope mode. It could be extended to study the transmission of activity across the neuromuscular junction by simultaneously recording the electrical activity in the attached gastrocnemius muscle. The Apple IIe pilot study introduced recording of action potentials in the human ulnar nerve for the medical students. The previous apparatus was not safe for human recordings and computer averaging techniques are needed for recording activity from sensory nerve fibres (electroneurograms ENG).Measurement of propagation in central pathways may also be possible in the human, using evoked cortical potentials.

Simulations are important in the area of membrane potentials and excitability where students have to deal with many difficult concepts.

Experiment 4: Properties of smooth muscle

Examines the control of the rhythmical contractions and overall tone of visceral smooth muscle of the rabbit in an organ bath. The importance of local environment, temperature and humoral agents are examined. Slow data acquisition using the chart recorder mode is required to measure the rhythmical contractions. A sensitive isotonic length transducer is needed. The computer could calculate the frequency of the underlying rhythm while recording the cyclical length changes.

Experiment 5: Haematology

Involves the analysis of the students own blood. No on-line data acquisition is required although a central computer would assist in the collation of class results. Results from some of the more modern analytical machines could be directly interface to the central machine. These could now include blood gas analysis.

Experiment 6: Cardiac muscle

Investigates the properties of cardiac muscle in the toad, relating the sequence of events and the coordination of contraction of the heart chambers. The experiments could be supplemented with the new equipment to include a pseudo-intracellular recording of ventricular action potentials (using a suction electrode) so permitting a better understanding of the electrical events associated with contraction.

Experiment 7: E.C.G. and heart sounds

Investigates the ECG (electrocardiogram) and PCG (phonocardiogram) in the human. Experiments are also performed with the exposed heart of the toad, so that the basis of the ECG recordings can be appreciated. These experiments would be considerably improved when two channels of recording can be made. The ECG and PCG can be simultaneously recorded without the confusion of superimposing their electrical signals in a single record. Vector cardiograms can also be generated from two simultaneous ECG recordings. The ability to record the ECG with a computer calculation of instantaneous heart rate will benefit many other experiments, e.g. exercise tests.Access to stored ECG's of showing normal population variations, such as with orientation of the heart, would assist the medical student's appreciation of the ECG. Some abnormal ECG recordings night also be valuable.

Experiment 8: Mammalian blood pressure

This experiment investigates the control of blood pressure in the anaesthetized rabbit. This experiment would be much more valuable if there were more than the current single recording of arterial blood pressure supplemented by calculations of heart rate. Simultaneous recording of venous blood pressure would allow students to better examine compensatory mechanisms. The ECG could also be measured. The experiment would need to be re-considered to take full advantage of the new apparatus which could also include the measurement of cardiac output by thermodilution techniques.

Experiment 9: Haemorrhage

This experiment investigates the cardiovascular compensatory responses to blood loss in the rabbit. This is an important experiment in the understanding of cardiovascular control and would be improved by the better monitoring of responses as outlined in Experiment 8.

Experiment 10: Human blood pressure and heart rate

Currently involves the measurement of human blood pressure and its variation with posture and exercise. The computer would be of little use in the measurement of blood pressure with cuff and stethoscope, but there is an important missing piece of information in this experiment, the variation of Cardiac Output. The computer will permit the calculation of cardiac output in all these situations so that a better understanding of cardiovascular compensation is possible. As well, during exercise, the beat to beat measurement of heart rate from a simultaneous recording of the ECG will permit a better understanding of recovery mechanisms. Simulations of cardiovascular function are particularly important, because of conceptual difficulties the students have with many of the interdependent variables influencing the circulatory system.

Experiment 11: Respiration

Part 1: Measurement of respiratory gases, breath-holding etc. There is no requirement for on-line data acquisition.

Part 2: Lung volumes and metabolic rate. Students need to monitor the flow of gas in and out of their lungs and collect samples for gas analysis. There are an inadequate number of conventional spirometers for this task at present, and the use of flowmeters connected to the computer would provide a more cost effective replacement.

Part 3: Exercise responses. Currently, students work in large groups collecting a variety of data manually supplemented by an ECG recording to calculate heart rate. It will be possible to have more students performing the experiments if the chart recorder directly monitors the respiratory and cardiovascular changes. Most importantly, the measurement of cardiac output changes will allow a better understanding of the cardiovascular and respiratory responses to a graded exercise challenge.

Experiment 12: Renal Physiology

Examines the human renal responses to a variety of stresses such as water and salt loading. There is no requirement of on-line data acquisition, but a central computer which could collate the students' extensive results within the various subgroups would considerably assist the staff in preparing for the lecture-discussion which occurs at the end of each practical class. It could also be useful for some more complex calculations.

Experiment 13: Gastric secretion

This experiment examines the modification of gastric secretions under various conditions. Some accumulation of class results with a single computer could be helpful.

Experiment 14: Body temperature control

This is a very important practical class for the students because of the significance of understanding temperature stresses and because it re-examines a wide range of physiological mechanisms involved in the integrated response to these stresses. The experiment would be considerably improved if cardiac output could be monitored, as well a having on-line recordings of ECG, respiratory flow, as well as the current recording of core temperature from the tympanic membrane via a thermistor probe.

Experiment 15: Sensory and motor functions in the C.N.S.

Current experiments use a limited number and range of apparatus to investigate sensory mechanisms and motor functions in the central nervous system. The use of computer based apparatus in the standard configuration with 2-4 equipped with signal generators to act as audiometers would considerably expand the range of experiments which could be offered.

Experiment 16: Structure and function in the C.N.S.

This is an orientation class for science students to provide them with an elementary understanding of the relationships of human brain structures during demonstrations and investigation of prepared material. No computer use is envisaged at present, although pre-recorded computer images demonstrating brain scan results could be useful together with 3-d rotatable representations of the location various brain structures.

Experiment 17: Endocrinology

At present, the only endocrinology practical class is an anatomical investigation of relevant structures in the rat. The new apparatus would permit students to investigate the action of anti-diuretic hormone on ion transport in the frog skin as a model for their understanding of kidney function.

Experiment 18: Salivary secretion

This experiment is for the dental students and examines the control of salivary secretion. Some accumulation of class data on a single computer is all that would be envisaged.

Interactive laser disc in Community Medicine

Trevor Lord
Department of Community Medicine
Monash University

Teaching Community Medicine requires a great deal of small group work. In order to provide time for this and still expose students to the necessary range of information a new approach was required. The Department developed a learning laboratory where the students could spend time working on learning packages. The packages have taken the form of video, computer problem solving, caromed, audio tapes and printed material. Interactive laser disc is the latest addition to the learning laboratory. Some 3000 slides including all the departments slides have been written onto a 30cm laser disc. Using an intelligent laser player and a touch screen monitor packages have been written using the laser disc as the material source. The department intends converting the software to EPROMS allowing free standing players and monitors to be set up without the need to have a computer attached.

Much of medical teaching is based around clinical situations requiring live patients, clinical slides or video. Using computers to mimic the clinical situations has been limited by this inability to provide the illustrations. Graphics have not solved this problem. Another problem is the large collections of clinical slides retained within the clinical departments.

Laser disc offers the opportunity to have a source of carefully selected clinical slides or video footage that the computer programs can then use to illustrate the clinical problems. Large numbers of slides can be stored on the one disc.

The Department of Community Medicine Monash University, has like many other departments, had its teaching resources stretched. The increasing pressure for Medicine Faculties to increase the community medicine components of their courses have added to this. There is also an increasing need to teach consulting and communication skills a role that has fallen to departments of community medicine. Teaching in this area requires high staff to student ratios yet the more basic content of general practice still has to be covered. For these reasons our department has been developing learning packages for some years. The students in our department spend almost 50% of their time in the department working in a learning laboratory going through the packages at their own pace. The enthusiasm for this kind of learning has encouraged us to continue in producing more packages and move into computer based packages.

Our Apple network based CALS have been in use for three years and include an assessment CAL that marks each student performing it and produces the scores. These clinical cases have always been hampered by the inability to illustrate them. In parallel with this has been caromed based clinical slides and video tapes of consultations with work through programs in booklet form.

The principle behind all the programs is that they are interactive with the emphasis on problem solving and not on presenting large quantities of factual information. The programs are kept to around 20 or 30 minutes completion time.

The laser disc technology offered us an ideal opportunity to combine all these into this exciting and fun to use medium. Our enthusiasm to use the medium has encouraged us to go ahead with out waiting to see which direction the technology is going to develop.

We have chosen the thirty centimetre laser discs because of the volume of material they can hold. Approximately 30,000 clinical slides can be stored on the disc or about 30 minutes of video footage. Any combination of slides or video can be used. The material for our first disc was prepared in South Australia and was sent overseas for production. We have taken a 10% share of this disc. This allowed us to load our departments collection of some 1,000 slides. We also obtained a first class selection of dermatology slides from a teaching Dermatologist, a collection of slides from Melbourne's Ear nose and Throat hospital and a collection of ophthalmology slides. A total of 3,000 slides.

Our choice of hardware was based on the direction that we wanted the packages to take. We wanted the packages to be simple to use ideally free of computers and keyboards. We wanted good interactive packages that could be easily altered or upgraded. We wanted the students to be able to select from a range of short interactive packages and easily start the package without supervision.

Two directions were open. The first was to use a simple laser player connected to a computer. Install a card in the computer that allowed the computer to drive the laser player to obtain the illustrations and produce the text for the interactive programs. This system required that a computer was constantly attached and that the student would need to drive the computer.The second option was to use the laservision technology and the Philips Ilvas authoring system.

Ilvas when combined with an intelligent laser player allows the program to be dropped into the player from the computer or from an EPROM in a plug in cartridge. The player can then be driven by a simple infra red hand held keyboard or a touch screen monitor. We were very impressed with the ease of use of the touch screen monitor which particularly suited our style of packages.

The hardware available at that stage was the Philips VP835/E Laser player and the Philips VP120 Touch)screen monitor. The cost of this approach was about $500 more than using the computer based card system with a plain monitor and simple laser player.

The Ilvas software runs on a IBM compatible machine using two floppy disc drives. It has its own operating system and is not currently available in a hard disc version. The second drive contains the data disk. It generates text in videotext format suitable for the monitors videotext decoder and has commands that allow full control of the player. The language is Allva which allows high level commands.

A supervisor allows control of the different modes of the program. The programs are made up in small simple fragments which are then linked. The text frames are generated separately and called up by the programs. The programs are compiled by Ilvas and can then be dumped to the 64k ram of the player. Once the program is complete and has been fully tested Ilvas manager can then be used to send the program to an EPROM burner for production of a chip which is then placed in a simple cartridge. The cartridge can then be labeled and available to the students in a fairly robust form.

Writing instructions for the touch screen is relatively simple. Screen co-ordinates are given for the positioning of the touch sensitive areas. Corresponding coloured boxes are then produced in the text frames. The boxes can be made to change colour when touched. Information as to choices made on the touch screen can be retained for scoring or later use in the program.

Control of the player from within the program is also powerful. Any frame can be called up from the disc within 0.5 secs. A series of frames e.g. video can be played forward, backward, slow or fast. Sections can be repeatedly replayed. Writing programs to utilize slides or video is relatively simple.

Integrating the videotext output and the laser disc output is also easy. Text can be over–layed over slide frames taken off the disk or over running video frames. The video text can be altered independently of the laser disc material so that a piece of video can be run for a while before superimposing videotext. Videotext can be displayed by itself. The text material can be held on screen for a period of time moving on automatically or sit awaiting a prompt from the touch screen. Text can be in a range of colours with varying backgrounds, limited graphics are available. Text can be made to flash or change colour.

The size of programs is limited by the 64k RAM of the player. We prefer short 20)30 minute programs and this limitation has not been a major handicap except when we have used a lot of text materiai. Programs that source most of the illustrations from the disc are not affected by this. Larger programs could be produced and written on the disc as part of disc production however this limits the flexibility of being able to change or upgrade the programs which is a serious limitation in the medical area. Ideally any large volume text material should be written on the disc as long as one is confident that it will stay in date. The advent of read)write laserdisc technology will resolve this problem.

From the learners view the face of these packages is very attractive. The complete system is a simple stand)alone player and touch monitor. The student selects the cartridge containing the program, plugs it in and switches on the player. The player boots from the cartridge and presents the information through the touch screen monitor. The learner touches the appropriate boxes to select choices. Clinical slides are taken from the disc to illustrate the problems. The presentation is rapid and colourful. The packages have a great deal of impact are fun to do and a very useful way of repeatedly presenting useful packages.

Our first packages were on ECG programs. They were surprisingly easy to produce. Lessons and programming ideas developed on one package can easily be copied and used in other packages. The on–screen editing of packages can be done on computer monitor or with the TV monitor allowing quick assessment of the presentation colours and methods.

A number of limitations are present. The inability to import files from a standard word processor. The screen editor is reasonably good but production of text could be enhanced if it could be performed on a wordprocessor and then moved into Ilvas. Another problem is not being able to use Ilvas from the hard disc. Reading and writing fragments to the floppy is slow and also hampers production.

Overall we are very pleased with the choice we have made. Given the goals that we had for the package design and presentation we feel the Philips system has met our needs very well. We are keen to start working with video footage. At present the Royal College of General Practitioners (U.K) are mastering a dedicated 30 cm disc for general practice. This will include video of consultations as well as slides. We are looking forward to obtaining this as a source disc and developing our own programs.

Monash is considering producing 30cm laser discs. Currently we are negotiating funding to produce a disc on Back-pain, communication skills and on AIDS. Once the discs are produced one can build an almost infinite number of programs using the source material. Communication skills will be the most challenging to produce in a learning package. However the ability to video a good counseling type consultation, run it slowly, freeze it at critical points, superimpose points and questions offers a unique opportunity to provide this type of learning.

The laser disc system will solve a number of problems in presenting our learning packages. The simplicity reduces the need for any supervision. The presentation ensures the students interest. The technology allows imaginative programs to be produced in areas that traditionally have required considerable staff input.

Trevor Lord, a medical graduate, spent four years in general practice developing educational interests through the Family Medicine Programme. Computing interests developed through the Royal Australian College of General Practitioners initially in the development of on)line computer assisted medical records. Secretary to the Colleges National Computer Committee, Victorian Computers in General Practice committee and co-ordinator of the RACGP National Computer Assisted Practice Project. Currently working in the Department of Community Medicine developing the departments computing role including production of patient management CALS, medical computing courses, medical computing research and assisting in the development of commercial medical record, prevention and health promotion software.

Lateral teaching software: A programmer's perspective.

Ralph S Sutherland & Benjamin K Selinger.
Chemistry Department
Australian National University

The microcomputer has opened up a new era in teaching tools and the time has come to assess its use. A project to develop MacFourier gave one of us (RSS), a computer programmer and a student, the opportunity to explore the situation from a variety of angles. This paper will cover some of the major design parameters that went into the development process. Along the way, specific examples of the principles involved will be illustrated.

User initiation and authorization.

One of the most important parameters was to keep the user in control. The user decides what process occurs next, and when a chain of processes can be stopped. This 'user driven mode' is especially evident in the software used on the Macintosh™ computers. It should be noted however that the choice of this mode of operation is the responsibility of the program, not the machine.

A popular method of implementing this style of programming is to use a dendritic program structure. This structure has a central waiting loop which can detect actions by the user, and a collection of subroutines that are selected in response to those actions. Additional cross links and returning routes between the various routines and the 'main loop' give the program its final behaviour pattern. Most of the running time is spent at the primary node or main event loop waiting for the user to act.

In a typical Macintosh application, each individual keystroke or mouse button push is intercepted individually by the main loop and passed to the appropriate handling routine, so that at any given moment the program is ready to respond to the user. In MacFourier a user can have just transformed a function, spotted an error in the transformed display and almost immediately proceed to edit the original function. This has the obvious advantages. It encourages semi-random exploration of the program's abilities without fear of losing everything because of an error. It means the program will not go through a procedure unless and until the user is both ready to, and has authorized the operation. Thus routines that are not wanted for a particular run need never be used.

Program restrictions on the user are largely replaced by user restrictions on the program. The computer is placed in the background, while problems and program solutions are brought forward. This lessens computerphobia.

Non-linear program structures, menus and more...

In the following examples some Macintosh terms are used, for which equivalent items and methods will be found on other modern micros.

Menus are a method of outlining the broad choices available to the user at any given

The Structure of MacFourier

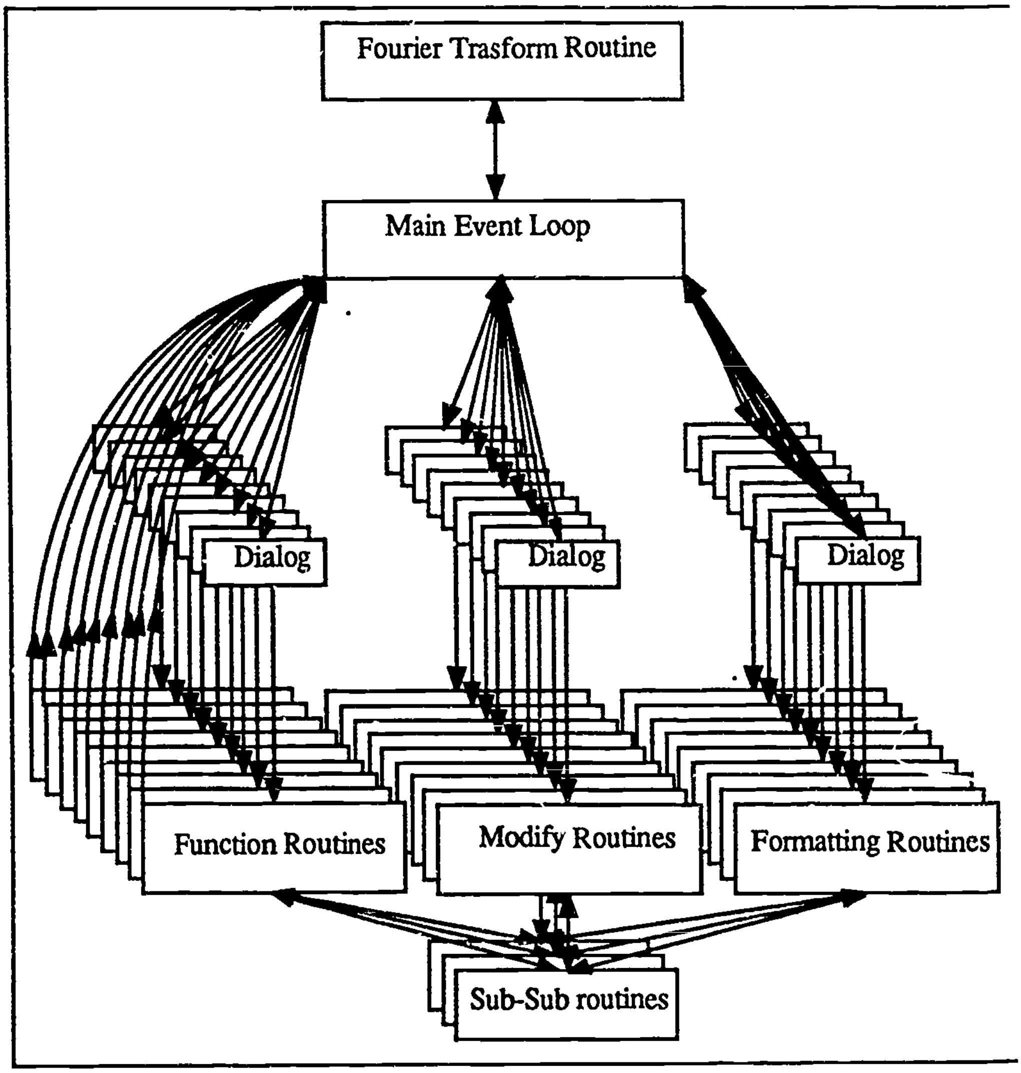


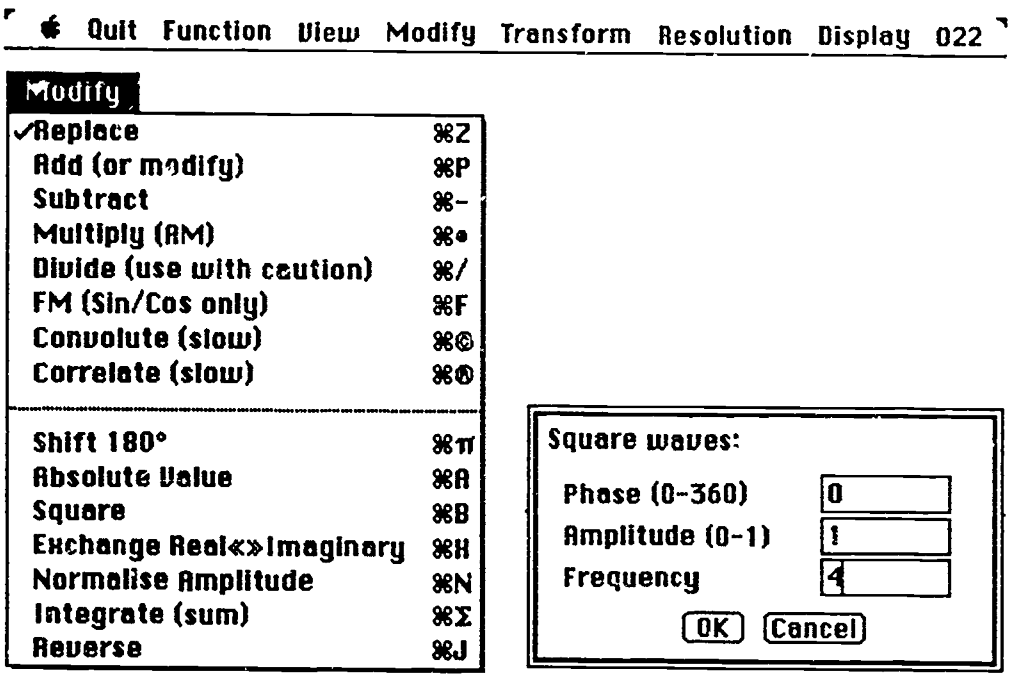
Figure 1: Non linear structure of the program (not all of the returns to the main loop are shown)...

time. They are enabled or disabled as required by the program to eliminate illogical or invalid options. The main event loop can detect a menu choice through monitoring a mouse event, and can then branch the program to the appropriate subroutine.

Further choices can be made with dialog boxes and buttons. They can be seen as buffers. The "CANCEL" button, e.g., allows the user to abort and return to the main loop. Other choices are offered as deemed desirable.

Relative values vs Absolutes.

When comparing the fields in which Fourier Transforms are used an immediate problem quickly arose as to whose conventions to use. One text outlined no less than seven different conventions in expressing the transform and its inverse. The values of



Square waves:

Phase (0-360) 0
Amplitude (0-1) 1
Frequency 4

Fig 2: Menubar and menu, dialog box and buttons.

the normalising coefficients was the main area of disagreement. We made the decision to avoid the problem altogether by not using an explicit normalisation. All displayed functions and transforms are normalised relative to an appropriate maximum and scaled to fit the screen. As a consequence labeled, graduated scales for the displaysare not used and there are no restrictions in the inputting of amplitudes.

A menu option can be used to scale down amplitudes for display. Dialog boxes make use of 'intuitive' ranges such as amplitudes of 0-1, and the centre of distributive functions can be placed on a 0-1 scale without using explicit channel numbers. This is just one of the steps taken towards simplifying operations and giving the user more time to think about the physical issues involved.

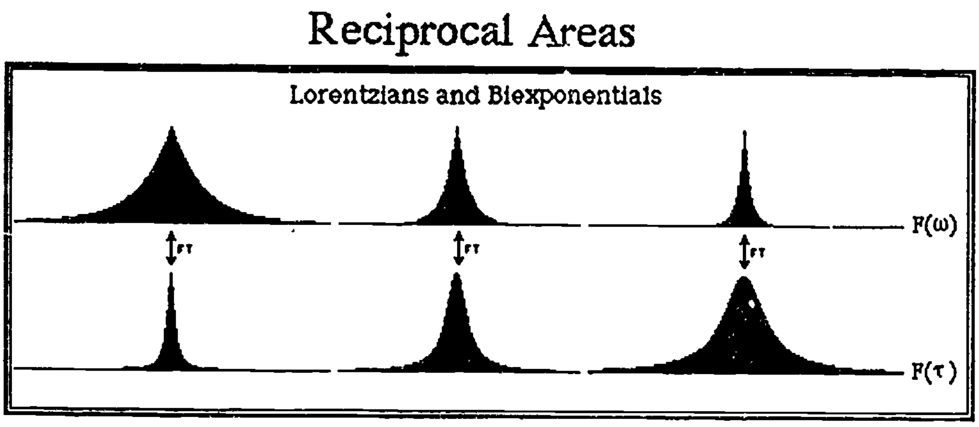


Fig 3: The Qualitative Uncertainty Principle, note the lack of absolute scales, yet the trend (and so the idea) is immediately obvious.

Levels of volunteered information, towards 'qualitative' programs

This trend towards a more qualitative approach became an important aspect of the design philosophy of MacFourier.

Visual feedback during processes

In order to allow the user to see what was happening at any given time, the modify and function operations can be seen evolving on the screen as they are carried out. Unfortunately the structure of the transform algorithm itself does not allow a similar approach tc be used. This leaves the transform operation as somewhat of a 'black box'. However the whole point of the program is to look at the properties of this 'box' rather than understand its particular algorithm.

Visual Feedback
Multiplication (a squarewave by an exponential)

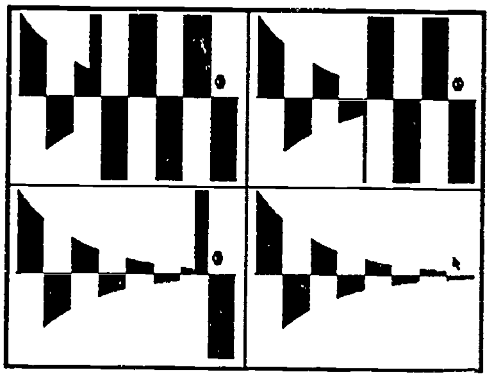


Fig 4: Visual Feedback. Simple operations are demistified by being displayed as they are being executed.

Allowance for creativity

Choices were made as to what basic functions should be included for the purpose of constructing functions and transforms. The final list of functions and operations included the modify menu allowed the construction of the simulations and problems found in all the reference texts. To construct some arbitary function seen in a text or otherwise desired requires some creative effort, the construction methods are not always obvious.

Active exploration of limitations of computer (especially digital) techniques

During development, one of the test functions used to test the algorithm was a square wave, which should theoretically give a purely imaginary transform. It doesn't! The 'error' can be tracked down to a fundamental problem of using a finite representation of the function and transform, i.e. the diff ence between a Fourier series and a Fourier integral.

Rather than trying to reduce the effect by increasing channel numbers or introducing sophisticated interpolation — smoothing algorithms etc, this 'error' was left in the program operation and emphasised in the teaching process as a fundamental property of any digital transform process.

General applicability to many fields, efficient use of programming resources

Fourier Transforms were selected as the first of the lateral teaching topics because they provide a general purpose tool common to many fields of study and therefore represented a teaching redundancy. In addition the usual mathematical approaches often did little for the student's understanding of the physical aspects that were the real learning objective. In addition, all the processes critically related to the Fourier Transform, such as correlation and convolution, could easily be linked. A major obstacle to be overcome was to provide a translation for the local dialects used to explain basically the same concepts in each of the different fields. Thus the spectroscopists' dispersion/absorbsion curve

A Weakness of Digital Techniques. Finite Representation error.

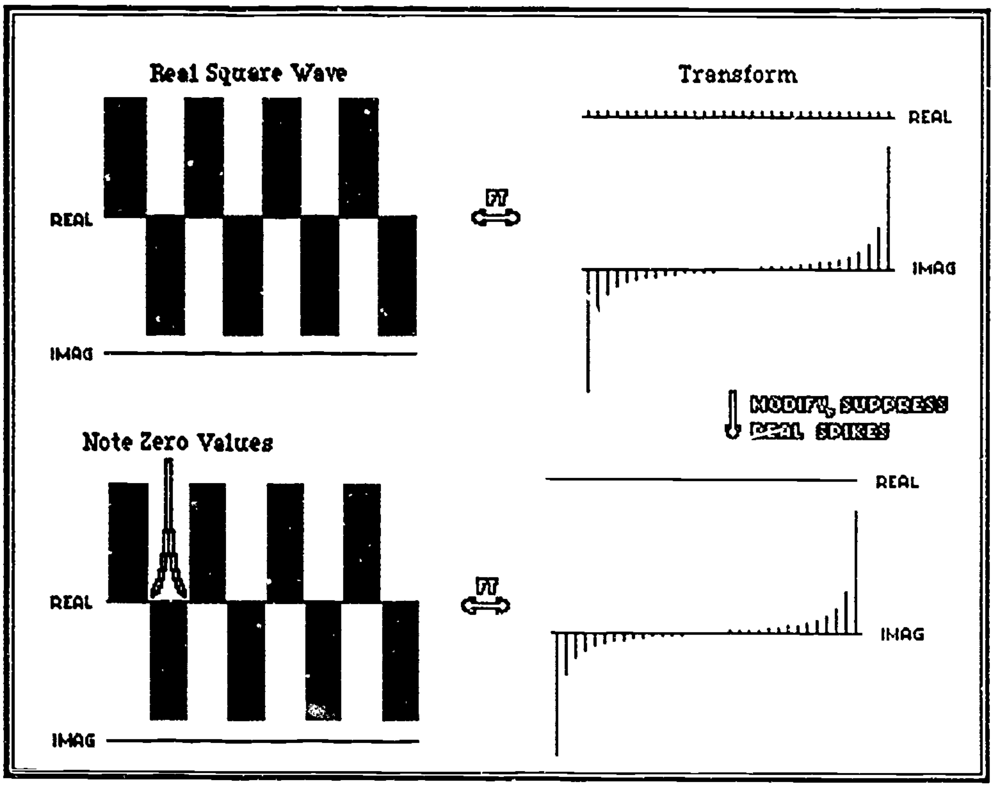


Fig5: The finite number of frequencies available to form a transform gives rise to the low fence of real spikes in the transform of a square wave. If the real component is removed by filtering etc, and the resulting spectrum is inverse transformed, a square wave is obtained that has zero 'crossover' values; these are not present in the original. As the contributing sinusoidal waves all have zero values, the zero crossing can only disappear in the limit of an infinte series.

is related to the sound-engineers log/log frequency response curves; the general slit function to the electronic system transfer function and so on. The similarities in different fields can be found just under the surface of the different conventions. One can go on. Any spectrum is a fourier transform. In physiology, the ear is treated a real-time continuous audio frequency fourier transformer. A lens is a two-dimensional transformer of the optical images. Bringing together disparate fields under a common rubric is our central theme.

There is nothing new in programming a computer to perform fourier transforms. The important thing here is a tool that can be used to mimic a tremendous variety of apparently unrelated problems/situations, and then to explore the relation-

FT-IR Spectroscopy:

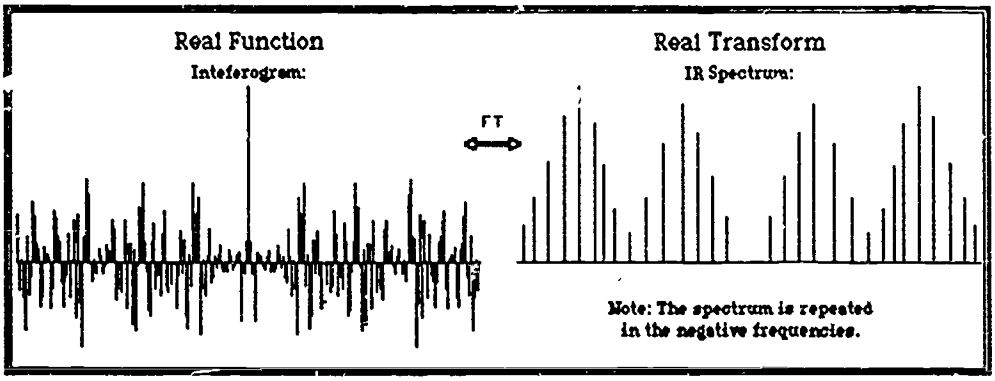


Fig 6: Any Spectrum is a Fourier transform.

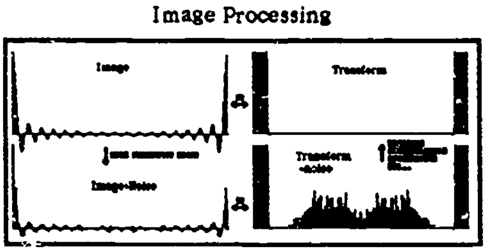


Fig 7: Fourier Transforms used in Image processing

ships that exist between them. The user can concentrate on the meanig of the transformations to his problem and not be concerned at this stage with a whole set of notational conventions or spend time understanding the transform algorithm itself.

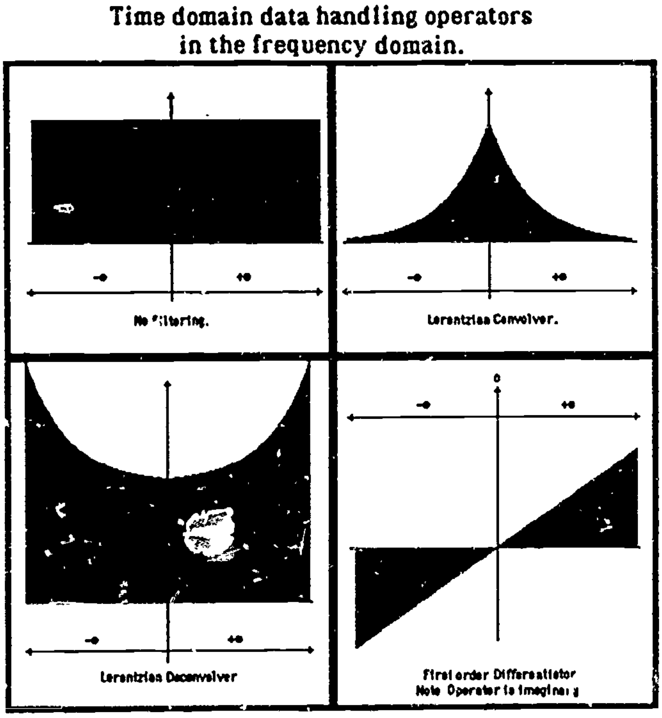


Fig 8: Many common, and apparently disparate data processing operations are in fact more closely related in the Fourier domain than in their original one. Differentiation (first order) of a function is equivalent to multiplying its transform by the *imaginary* ramp function, whereas frequency filtering can be achieved by multiplication by similar *real* ramps. Filtering can be related to convolution or deconvolution.

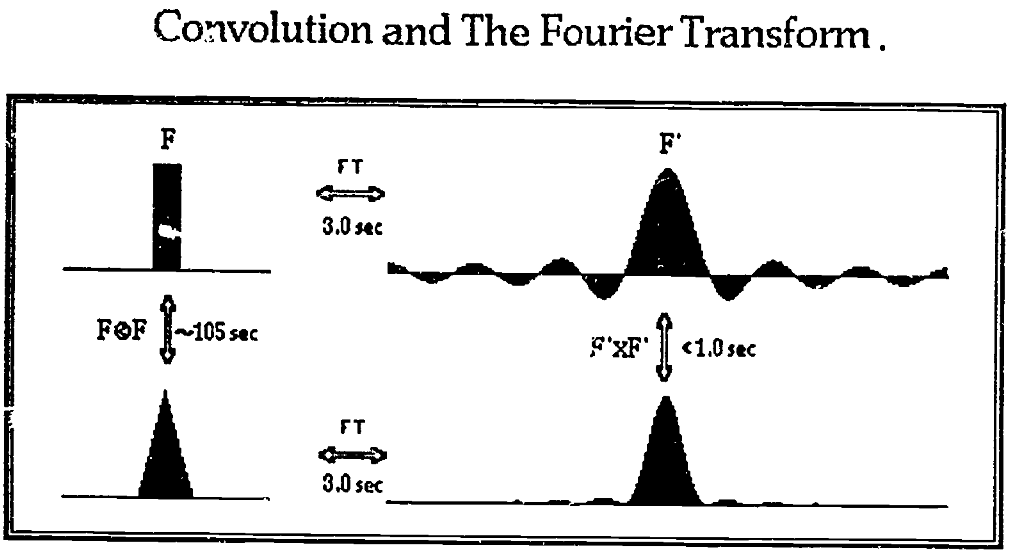
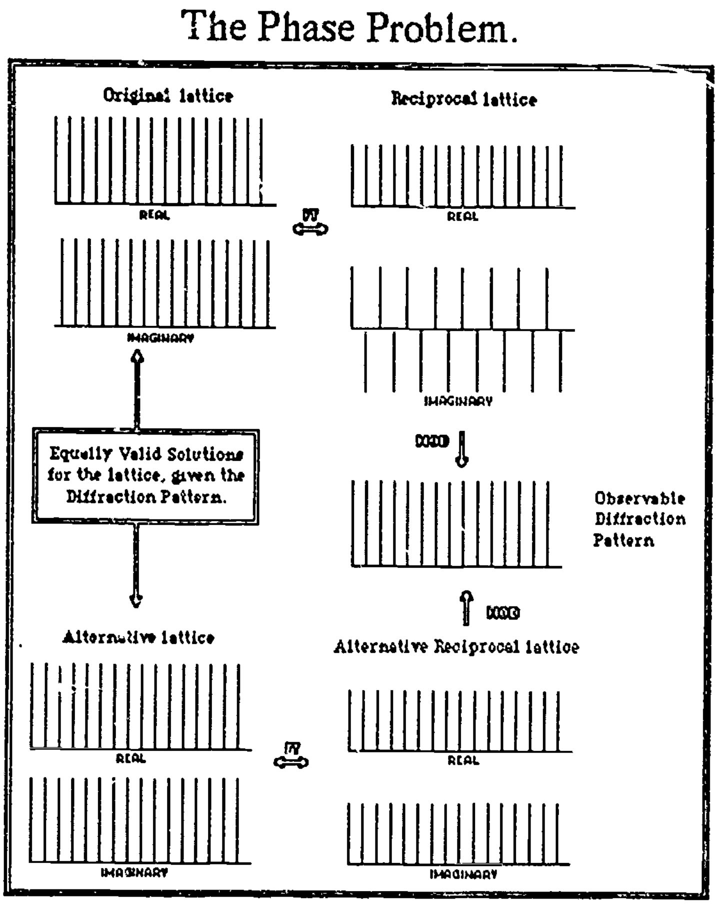


Fig 9: The nature of convolution can be illuminated by seeing the operation as the multiplication in transform space. The lateral design of MacFourier allows a user to 'enter this cycle at any point.



The Phase Problem.
Original lattice
Reciprocal lattice
REAL
FT
IMAGINARY
REAL
IMAGINARY
Equally Valid Solutions for the lattice, given the Diffraction Pattern.
Observable Diffraction Pattern
Alternative lattice
Alternative Reciprocal lattice
REAL
FT
IMAGINARY
REAL
IMAGINARY

Fig 10: The relationship between Fourier transforms and the crytallographic diffraction phase problem. One can only observe the modulus of the complex diffraction pattern (Fourier transform of the lattice) and so several alternative lattices are possible given one diffraction pattern. Here only one alternative is shown but others are possible.

Acknowledgement

The program development was funded under a Faculty Research Grant to B.K.S. for the topic 'A Lateral Approach to Tertiary Science'.

Interactive audio... A challenge or companion to interactive video ?

N.A. Shaw
Footscray Institute of Technology

In tertiary education there have recently been many advances, both in terms of hardware and software, in the style and application of new technology to the teaching process. This paper describes some recent experiences with the use of "interactive audio" techniques, and endeavours to show why this particular technology has an important niche in the spectrum of alternative teaching methods.

In addition, this paper compares interactive audio with its companion technology, interactive video. Whereas both of these technologies have the potential to enrich the educational process, the special attributes of each can only be fully appreciated when their appropriate *role* in the learning environment has been carefully determined and these features matched to specific teaching tasks.

To provide a substantive framework against which to compare these two interactive technologies, a context for learning is set by classifying both methodologies in an overall scheme of "intervention strategies" in teaching.

Intervention strategies in teaching

It would sometimes appear that there is a tendency for computer aided learning materials to be created with regard to the attributes of the technology rather than with reference to how and where the system fits into the overall educational process. To assist in developing an overview or framework, Sillitoe and Gorman (1987) have adapted the Morrill Cube (after Morrill *et. al.*, 1974), to illustrate that any successful intervention in student learning and welfare must consider these three major components:

a. the Target of the intervention strategy

b. the Purpose of the intervention strategy, and

c. the Methodology of the intervention.

By using this model, the technologies of interactive audio and interactive video can be located in the spectrum of possible intervention strategies. Furthermore, each of the three dimensions of this model has several parts, allowing further detail to be appreciated during a comparative study. This allows us to clearly illustrate the role and specific advantages of one technique over another in each situation.

According to this classification, both interactive audio and interactive video are interventions which have the following characteristics:

a. The *target* can be individual students or small groups of students.

b. The *purpose* a can be any one of remedial or crisis intervention, preventative intervention, or developmental or furthering intervention.

c. The *method* of intervention belongs to the category of educational technology.

By highlighting these separate elements of the intervention process, it is possible to focus upon the special characteristics of the intervention strategy which most adequately address the situation. For example, by observing that the technologies are directed at individual students or small groups of students, staff can allow an investigation of personal learning styles to influence the choice of delivery mode. Programs which are designed for delivery to a mass audience would not share this characteristic.

The role of educational technology

Within the category of "educational technology" there is a broad range of teaching styles, most of which are machine dependent and require little commitment of human resources at the stage of actual lesson delivery. However, by further subdividing educational technology into the categories of interactive and non–interactive, (see Figure 1), we can identify those teaching styles which use computers for *responding* to student input and for *controlling the delivery of materials,* from the others.

Figure 1: Two categories of educational technology.

	Non–interactive	Interactive
Examples:	Slide–Tape Shows Videotape	Computer Aided Learning

Finally, within that group of educational tec[illegible]ogies classified above as "interactive", there is a group of techniques which can be represented as in Figure 2. As can be appreciated from this diagram, when presenting learning material to students using these techniques each one depends to some extent upon visual stimuli (use of the monitor), while others may also involve the use of audio stimuli and enhanced visual presentation such as full colour video.

The following brief descriptio[illegible] identify the main components in this range of interactive techniques:

1. *Text:* The simplest style merely displays text on the computer monitor, and many of the early CAL lessons operated at this level. The introduction of commercially available packages such as "Storyboard" have enhanced this style by offering special effects, including a variety of intriguing ways of wiping or dissolving text on the screen and a variety of type fonts.

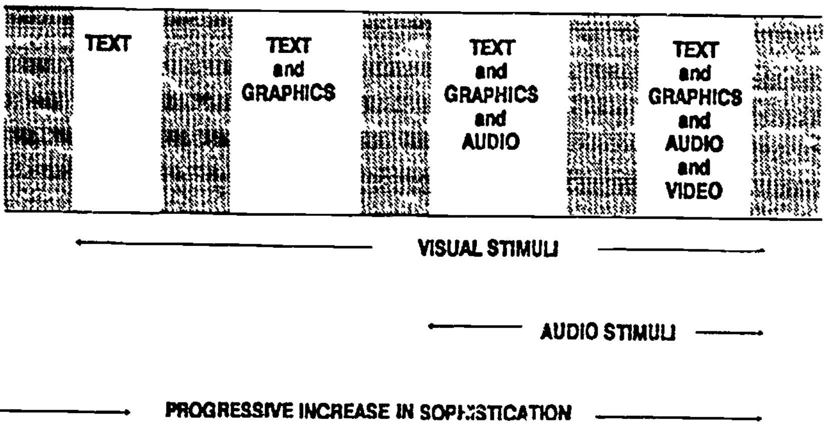
2. *Text and Graphics:* Adding graphics, particularly animated graphics, to a text–based lesson can further improve the presentation of materials as well as increase the level of student interest and alertness. Perhaps the majority of CAL programs in education today are operating at this level.

3. *Text, Graphics and Audio:* This is the domain of interactive audio, where text and graphics (or digitized pictures) can be displayed on the monitor simultaneously with the sound of voices, music, and other special sound effects. The added audio dimension offers great scope for teaching those concepts and skills which depend critically upon sound for full communication.

4. *Text, Graphics, Audio and Video:* This part of the spectrum refers specifically to interactive video, where selected portions of video can be shown to students under program control. The ability to superimpose text and graphics as overlays on the video as well as having full sound facilities, offers the CAL designer what appears to be the ultimate set of tools for lesson creation.

It is important to appreciate that in the above discussion, components of the spectrum were arranged in an order of increased degree of sophistication involved

Figure 2


TEXT
TEXT and GRAPHICS
TEXT and GRAPHICS and AUDIO
TEXT and GRAPHICS and AUDIO and VIDEO
VISUAL STIMULI
AUDIO STIMULI
PROGRESSIVE INCREASE IN SOPHISTICATION

in using the technologies. With progressively increased sophistication comes an increased degree of complexity and higher costs in both materials development and the type of peripherals needed for lesson delivery. Considerations of cost, effectiveness, preparation effort, and complexity are all critical elements which will be used later to compare interactive audio with interactive video.

Having identified the range of CAL techniques available, the next step is *matching* a specific teaching task to the appropriate choice of educational technology. Knowing the special attributes of each technology is most important when making a suitable choice. The following section describes interactive audio in some detail, identifying those special features which justify its place in the spectrum of intervention strategies.

The technology of interactive audio

Any device which can record and reproduce audio information such as voice, music, or sound effects, and at the same time have all its functions controlled by a computer, is likely to be classified as "interactive audio". A typical program can display normal text and animated graphics on the computer monitor, whilst synchronized sounds are reproduced as desired.

The method by which the audio information is stored may be either analogue or digital. For example, an analogue technique could be an audio tape recorder/player which stores the data on normal audio cassette, although only a few commercially available players are capable of computer control. In this context, "control" refers to the ability of the computer to perform certain tasks with the tape such as:

- searching quickly for any desired position on the tape,
- playing or recording on the tape whilst simultaneously monitoring the position, and
- controlling the volume.

The two common digital techniques of storing audio are:

(i) compact disc, (which is laser technology), and
(ii) floppy or hard disc (normal magnetic media).

Similarly the sounds stored digitally on a floppy disc can be retrieved quickly and reconstituted back into high quality audio. Experiments have shown that recording 42 minutes of high fidelity sound consumes about 10 megabytes of memory. If an application could tolerate lower quality recording, then up to nine hours of sound can be stored in the same 10 megabytes. Whatever the audio storage medium therefore, an interactive audio system must have the ability to store and quickly retrieve such sounds under computer control.

Special attributes of interactive audio

As indicated earlier, the selection of one style of educational technology over another is determined not only by the nature of the particular task in hand, but also by the special attributes which each style can offer. Too often we see one particular technology being forced to perform a Procrustean task better suited to another teaching method.

So what are the particular properties which make interactive audio an attractive, advantageous alternative in certain teaching applications? Some of the most important aspects are as follows:

a. The *interactive* nature of the system is an inherent part of any computer based lesson. Appropriate design of the program allows students to control their own *rate* of progress with the lesson, as well as their own *path* through the material. This places students clearly at the centre of the learning process, an important attribute which is common to most CAL lessons.

b. *Audio:* In cases where special sounds are central to the concept or skill which students are seeking to learn, interactive audio is particularly useful. As will be discussed later for example, describing the relationship between Korotkoff sounds and blood pressure readings on a sphygmomanometer cannot be effectively conveyed without fully reproducing the genuine sounds. In addition, the teaching of nursing students to recognize foetal heart beats is also greatly enhanced if these sounds are actually used as part of the lesson. Even voice communication itself, with the appropriate emphasis and intonation, is potentially more effective than relying on students to read a computer screen full of text. It is an emphasis on this audio aspect of the system which helps to set it apart from any other learning technique.

c. *Visual reinforcement* of the audio concepts is achieved by simultaneously displaying text and graphics on the monitor. As the voice explains certain aspects of the lesson, key words (or whole sentences) can appear on the screen, thereby supporting and adding constructively to the learning process.

   Animated graphics can also be used to enhance a simulation. For example, a graphic representing a sphygmomanometer can be animated so that mercury in the column appears to be dropping. Whilst the "mercury" is dropping, synchronized blood pressure sounds are reproduced to complete a realistic simulation.

   With the introduction of digital scanning devices, a photograph can be copied onto floppy disc, incorporated into the lesson, then later displayed on the monitor with additional text, graphics, and audio enhancements.

d. *Costs:* As in any educational technology there are production costs involved with the source materials and programs as well as delivery costs in presenting the product to students. Fortunately, interactive audio is not an expensive medium in which to invest, either for production or delivery. Whilst interactive audio relies upon some device for its source of audio, whether cassette tape, compact disc or a digital audio interface, the cost of adding an appropriate interface to transform a normal personal computer into an interactive audio system is less than the cost of the computer itself.

e. *Ease of production:* Unlike some other new technologies such as videodisc, interactive audio allows complete lecturer control of the product, as well as requiring only a relatively short production time. Experience with the medium has shown that feedback from staff and students who use the program can be readily incorporated into the lesson. This can be achieved either by modifying the audio tape with simple recording equipment, or by creating new digital audio files on floppy disc.

After field trials which may involve further development and refinement of the program, the product can be converted with confidence to a permanent, reliable, and fast–access medium such as a compact disc.

Specific examples of interactive audio

The following paragraphs briefly describe the elements of three projects which make use of this technology.

(i) *"Measuring Blood Pressure"* This program, which has been designed for first year student nurses, has the following four main parts:
- identification of the Korotkoff sounds (especially those associated with the systolic and diastolic blood pressures).
- reading the sphygmomanometer scale whilst listening to the specific Korotkoff sounds, thereby performing a blood pressure measurement.
- experiencing those mechanical factors which influence blood pressure readings, and
- hearing a variety of Korotkoff sounds from people of different age groups and health status.

At least three of these four parts depend extensively upon audio as the essential means of conveying information, thereby justifying this choice of technology over other techniques.

(ii) *"The Foetal Heart"* is another interactive audio program with a medical education theme which depends extensively on specially recorded sounds. Again the program is aimed at undergraduate nurses, but it is also appropriate as an introduction for students of mid–wifery courses. The program has three main parts:
- locating the foetus in the uterus, and identifying the particular "lie",
- locating the foetal heart, using a knowledge of the foetal lie in conjunction with a knowledge of the characteristic sounds created by the foetal heart, placenta, and other mother–body noises,
- counting the foetal heart rate, and determining if the foetus is healthy.

It will be appreciated that, for ethical reasons, it is not reasonable to allow teams of undergraduate student nurses to examine and prod the abdomen of selected pregnant women. However, use of this interactive audio program can simulate a great deal of the clinical situation by using genuine sounds recorded with help from fully trained medical staff.

(iii) *Spelling:* The third example to be briefly described is an interactive audio program designed to help those tertiary students who have significant problems with spelling. One technique is to read a sentence containing the particular word and at the same time display the sentence on the monitor. In this example, the aural, visual and in–context clues are all important parts of the task. One of the greatest problems encountered by students is converting a lecturer's comments accurately into the written word. As can be appreciated, without an audio component to this basic literacy program, the strategy would loose much of its impact and relevance..

Criteria for selecting Interactive audio

In common with most computer based learning projects, there is a considerable commitment in terms of time and resources in developing each lesson. Therefore when academic staff submit a proposal or request for the development of a technology based lesson, considerable effort is spent on matching these requirements with an appropriate choice of educational technology, ascertaining as clearly as possible the learning skills involved and detailing the desired outcomes.

Projects selected for development with the interactive audio tool have generally satisfied both the above philosophy and the following criteria:

1. The subject matter must have a critical dependence upon aural cues and special sounds for conveying the desi ed knowledge or skill, ensuring that only those projects considered "appropriate" for interactive audio are developed with this tool.

2. Individual or small group attention is required to cater for a large range of student backgrounds and abilities.

3. The project must be of fundamental importance to a curriculum offered on this campus. Ideally the project will be of value not only a large number of students on one campus, but hopefully in similar courses offered at other educational institutions.

The Production Team

A successful model for the development of computer aided learning materials has been proposed by Bork (1982, 1984) and adopted here as the production process. In essence, the approach encourages the use of a multi–discipline team on every project, with the assumption that the integrated effect of a group of specialists contributing to the project will result in a better product than the concerted efforts of an individual. Authoring programs, which are designed to encourage any or all teaching staff to create their own computer aided lessons, tend to discourage this team approach.

Although only a few interactive audio programs have so far been produced at Footscray Institute, already a scheme has evolved whereby the resources of a variety of people are used in the production process. Subject expertise is provided by the teacher who is ultimately responsible for the accuracy of the content. Special input comes from an instructional designer, a computer programmer for developing special effects, a script writer, graphic artist, and professionally trained voices recorded in broadcast studios.

Production time for a program is typically about five weeks. However several projects are concurrently being prepared and are all at different stages of development. Given appropriate conditions, it appears that one new program could emerge from the process about every 10 days.

Evaluation

Production of an interactive audio program is only one of the phases involving this team. Academic staff and the project co–ordinator are also committed to the process of delivering this material to students, evaluating the effectiveness of the project, and performing any program maintenance recommended during the evaluation. The interactive audio program "Measuring Blood Pressure" has so far been field tested with a significant group of students (150 first year nurses), and the "Foetal Heart" program is still under evaluation. Academic staff and students have reported successful use of the technology in the strongest positive terms, and a few suggested improvements have since been incorporated into the program.

Nine academic staff who participated in the evaluation phase reported the following observations:

(i) The time allocated for teaching assessment of blood pressure was greatly reduced.

(ii) Almost all students were able to identify the Korotkoff sounds required for blood pressure measurement when taking their first blood pressure reading. Frustration with learning the concept and acquiring the associated motor skill was greatly reduced.

(iii) Students enjoyed the interactive way of learning and the process became an interesting, non threatening way of learning an important psychomotor skill.

(iv) Co-ordinating students so that they were able to use the program immediately prior to practice in the skills laboratory was difficult, due mainly to an insufficient number of interactive audio delivery stations.

Student reactions to the program were also solicited, and almost all comments were extremely positive. The only negative response came from one mature age female student who stated that she "hated computers" and complained that she "felt disadvantaged from her peers because of her lack of experience with computers".

One class of twenty-two students was used as a control group in the evaluation. These students were asked to attend the skills laboratory in the traditional way, but afterwards were given the opportunity to use the interactive audio program. Comments from the control group covered the following points:

- Some wished that they had used the program prior to the skills laboratory.
- Some only understood the correlation between Korotkoff sounds and measurement with the sphygmomanometer after using the program, and
- Some heard for the first time the Korotkoff sounds which they had failed to hear or recognize during the laboratory session.

Since using the program with 150 student nurses, an additional "computer managed learning" component has been added. Each student now logs on with a unique identification code, and individual achievements with the program are recorded.

At the end of the lesson, students are invited to record their own comments. Using the the student's identification code, the tape recorder seeks a position on the tape which has been allocated to each student and the student is prompted to record comments for a maximum time of 60 seconds. These comments are recorded without effecting the audio tracks which contain material used in the lesson. This means of communication with the teacher, together with the automatically logged achievements, now provides a much more sophisticated way of analysing student reactions and performances.

Overall, both staff and student responses to the interactive audio program "Measuring Blood Pressure" were very encouraging. Based on this first level of experience with the tool, it can be claimed that interactive audio is a potentially useful technology, particularly for teaching skills and concepts which involve some auditory stimulus.

Comparing Interactive audio with Interactive Video:

Throughout this paper reference has occasionally been made to the companion technology, "interactive video". As a versatile, creative, and effective medium, videodisc has captured the imagination and enthusiasm of people in education, training, industry and commerce in many countries throughout the world. This is an appropriate place to compare the two technologies according to some specific criteria.

It should be acknowledged that each of these two technologies has similar but different attributes, and care should be taken to selectively apply the appropriate technology to the particular teaching task. In the discussion below, we have made a comparison of the two approaches assuming a hypothetical case where both appear to be equally suited to a given task.

a. Costs: Interactive video is an expensive technology, in terms of (i) the size of a production team, (ii) facilities and time needed for production, and the (iii) equipment needed for delivery of materials to students. It is estimated here that the production costs of a 12 inch videodisc project (involving considerable amounts of original video shooting) would be about 30 times greater than a corresponding interactive audio project.

b. Ease of maintenance and development: Once a videodisc has been pressed, its contents cannot be changed. If application of the disc in the teaching task reveals some design problems, the controlling computer program has to be revised to accommodate the error, or alternatively, another videodisc must be pressed with the errors corrected.

   With interactive audio, the project can be field-tested in the classroom using either analogue tape or digital (magnetic) format for the source of audio. Improvements and modifications to the content can be made as a result of the trials, and the medium need only become fixed when the final compact disc is pressed.

c. Ease of Production: The level of complexity of a videodisc production, particularly one for educational purposes, can result in excessive lead times. For example the videodisc produced in South Australia for teaching English as a Second Language ("The Aussie Barbecue") has been more than 12 months in production. On the other hand, interactive audio projects are considerable less complicated, and small production teams can develop programs within a few weeks.

d. Effectiveness and appeal: If the task requires the display of visual images representing the dynamic real-world, then, as suggested earlier, video is the natural choice. However, if the task can be adequately described with computer graphics (including still pictures), then an interactive audio program could be an attractive, effective and low–cost alternative solution.

The learning effectiveness of each mode of presentation could be a suitable topic for future research, especially if such a study incorporated an analysis of the different learning styles of students and the different modes of assimilation of information.

e. Versatile applications: A videodisc produced for one task can be used under different program control for new purposes. This versatility of the medium means that greater value can be extracted from the videodisc than originally intended, although this "re–purposing" usually makes us of only the visual material stored on disc. However, with interactive audio programs, there is less likelihood that materials produced for one program will be of any value or relevance to another.

f. Dual Applications: When a normal videodisc player is used to recall and display a single picture or still frame, there is usually no associated audio. However, if an interactive audio system is simultaneously under the same program control, then it is possible to use natural voice and other sound effects to complement the visuals coming from the videodisc.

Conclusion

Interactive audio is an effective teaching process which has an appropriate place in the spectrum of alternative teaching methods, and it is suggested that attention should be given to the special attributes of interactive audio which have been described in this paper. The examples which have been given from our own program development area illustrate how matching these features to specific teaching tasks can result in efficient and effective computer aided learning.

This paper has also emphasized that several criteria can and should be used in selecting an appropriate educational technology to apply to a specific task. Such selection criteria can include the effectiveness of the technology and how it relates to the different learning styles of students, as well as the flexibility, versatility, simplicity and cost of the production processes.

The work reported here also suggests that interactive audio performs well against most of these criteria, and represents a real alternative to its companion technology, interactive video, in a number of specific learning situations. Additionally, program designers should note that interactive audio and interactive video can complement one another if a situation could benefit from simultaneous application where an independent, flexible audio source is used with stored videodisc images.

The impact and importance of interactive audio in computer aided learning, especially at tertiary level, is still to be fully recognized. In those instances where the specific attributes of interactive audio are matched to both the learning task and to the learning style of students, this technology is strongly recommended to educators in the field of computer aided learning.

Acknowledgements

The author gratefully acknowledges that a grant for the "Disadvantaged Students Project" from the Commonwealth Tertiary Education Commission was used to acquire equipment and develop interactive audio programs. Footscray Institute of

Technology has also contributed to this development by creating a special full time academic position for developing technology-based educational materials.

References

Bork A., (1982). University Learning Centres and Computer Based Learning, *JCST*, 12(2), 81–86.

Morrill, W.H., Detting, E.R.; and Hurst, J.C., (1974). Dimensions of Counsellor Functioning. *Personnel and Guidance Journal*, 52, 354–359.

Sillitoe J. and Gorman, J., (1987). The Morrill Cube as a Theoretical Underpinning of Participation and Equity Initiatives. Paper presented at the HERDSA Conference, September, Perth.

Applications of interactive video

S. M. Byard and N. A. Shaw
Footscray Institute of Technology,

Language competency is a significant and growing problem amongst students at the Footscray Institute campus, and traditional methods of instruction appear to require supplementation in order that increasing numbers of students can be afforded the necessary assistance. This paper discusses one initiative which has been mounted to address this problem using interactive video instruction. The experiences gained during the introduction and use of this medium on a large scale are presented together with some student reactions to the trial.

There is a high proportion of students at Footscray Institute who are recent immigrants. This is, in part, due to the location of the Institute in the western region of Melbourne which is a traditional settlement area, and these people naturally look to FIT as the major tertiary Institution in this region of Melbourne to provide tertiary education for themselves and their children. Many of these students have been educated to tertiary or senior secondary level in a non-English language, and have achieved a satisfactory grounding in cognitive skills such as mathematics, engineering and science areas. However, these students are "underprepared" for tertiary education in Australia because their mother tongue is not English. In addition, other students may have had the majority of their secondary education in Australia, but live with families who have a first language other than English.

As a result, there is a considerable fraction of our students who have difficulty in appropriating the codes and nuances of language required of tertiary students. For example, many of our students have difficulties with such notions as "tone, relevance and jargon", the distinction between inductive and deductive argument is rarely understood, and the ability to handle simple and complex sentence structure, especially for those who rate themselves as weaker than the average student, seems clearly to lie outside their language repertoire. The Institute has been conscious of difficulties encountered by these students, and through the Institute Language Policy Committee, the Department of Humanities and the Educational Development Department, has been exploring a range of strategies to address this problem.

As part of these initiatives, interactive video delivery systems acquired under a grant from the Commonwealth Tertiary Education Commission for increasing participation in tertiary education by disadvantaged students, have been introduced into the regular teaching programme on a trial basis. Featuring the commercially available programme "Writing For Results" designed by K. Stuart and D. Limon, this presentation technique was integrated with the Written and Oral Communication subject taken by students of science and engineering.

Interactive video systems have been the subject of previous seminars at this and other conferences. We present this paper as a report of the trialling of this technique in our classrooms, and claim that this experience has enabled a better appreciation of this mode of instruction to be gained by members of teaching staff, and has enabled us to observe the difficulties likely to be found by other educational and industrial organizations seeking to implement such computer-based instructional schemes.

The delivery system and software.

Hardware involved in presenting the programmes to students consisted of a personal (compatible) computer with hard disc, a video disc player, colour monitor with fast blanking, and the MIC-2000 controlling interface card. This combination offered the facility of overlaying text and graphics information generated by the computer with video images coming from the video disc. Software consisted of controlling programmes (together with certain text and graphics overlays) stored on floppy or hard disc, and a set of corresponding videodiscs.

"Writing For Results" was produced by Stuart and Limon for the London based company of Interactive Information Systems (IIS), and marketed in a PAL video disc format. This training course is marketed as being "...designed to help you write more effective business letters and memos". It consists of four video disc modules entitled "The Basics, Planning Your Writing, Writing with Style, and When the Going Gets Tough", each structured to take approximately an hour to complete. These provide a comprehensive and sequential course, although each module is also self-contained.

The educational rationale for this approach.

The Humanities Department of the Institute considered Stuart and Limon's material suitable for the Written and Oral Communications course since the stated objectives of the IIS course focused particularly on the need to:

a. understand the basic elements of effective writing in organizational settings, and

b. apply a planned or disciplined approach to writing.

Clearly, in the light of our earlier comments regarding the background of many of the students at FIT, these objectives integrate well with the aims of a general literacy course at a technological institution.

Although a student text had been previously set for this subject, our traditional approach to teaching had also emphasised the use of "in-basket" techniques, real life exercises and audio-visual material to provide authentic contexts for students working on language development. Thus "Writing For Results" could be seen as a complement to existing curriculum materials. In addition however, interactive video seemed to offer us the possibility of extending these strategies in the letter/memo section of the course by allowing students to obtain repetitive reinforcement of sound principles of effective writing in a collaborative context which was not unduly teacher-intensive.

Consequently, it was agreed by the relevant standing Committee of the Humanities Department to use "Writing For Results" during the first five weeks of Semester One 1987 when students were to be working on an outline of communication theory and a unit on letters and memos.

The design of the project

We had initially planned that the interactive video modules would replace conventional instruction for one hour a week over a four-week period for a selected cohort of students. This proposal was contingent upon suitable supervised accommodation for the equipment and students being available within the Humanities building, but this proved to be impossible to organise with relevant teaching space and staff being unprecedentedly loaded during 1987.

Furthermore, because of the heavy timetable for students in Science/Engineering it became clear that it would be difficult to obtain sufficient free hours in a four week period for the one hundred (plus) students to make full use of the modules outside of timetabled hours. Within the scheduled class times, even using both delivery systems, it was evident that a one hour time slot was not enough time for most students to work through a module. Subsequent experience in use of this equipment in the Educational Development Department at the Institute now indicates that most students take six to eight hours to use all four modules, even when working in small groups.

With these problems in mind, and because of the added observation that many of the students seemed so keen to experience the material, it was decided that we would encourage all students to work through the modules. We decided that:

a. all students in designated classes were to be timetabled for the use of the material in groups of five while they simultaneously followed the prescribed course of instruction, and

b. all students were to be asked to compare the interactive video-based learning with the other mode and to complete a questionnaire at the end of the four-week sequence of study.

Students involved in the project

Initially, 105 students were identified as participants in this pilot scheme. The undergraduates enlisted for the trial were in six groups of full-time day students: five groups from first year Computer Technology and Digital Electronics and Computing, and one group of second year students in Building Engineering. One further group of part-time evening students in the second year of an Occupational Health and Safety Associate Diploma withdrew immediately because of timetable constraints. However, the change in project design discussed above caused an added reduction in the number of groups participating from seven to three, reducing the student population from 105 to 45. Further comment on this reduction in groups will be made later with the intention of highlighting some pragmatic difficulties regarding the introduction of new technology into existing discipline areas.

Some practical considerations of the method of delivery.

As with all valuable equipment, there was a necessary requirement for supervised space in which to house the interactive video system. In addition, we found the need for close supervision of some of the students. An attempt was made by staff to check with students at the beginning and end of each module and at regular intervals within the hour, which, although adding to the time commitment to the project, gave the teacher a flexible opportunity to monitor each student's grasp of the material. It also had the benefit of checking whether one student was monopolizing the keyboard, or observing if students were working in languages other than English contrary to instructions.

The former problem arose since the modules were designed for individual use, and naturally encouraged single-user responses. Because the majority of the students were in the first year of their tertiary course, the expectation that they should adapt to small group work smoothly and without difficulty is perhaps a little unfair. In addition, the physical requirement for groups of students to congregate in a confined area in front of the monitor led occasionally to what one student referred to as a "tendency toward group larrickinism", requiring a certain amount of social control by the lecturer.

Whereas none of these problems are specifically due to the equipment or material involved in interactive video, it is nevertheless an important area of consideration when planning the introduction of new techniques with large groups of relatively unsophisticated students.

Reactions of staff involved in using the system.

The marked reduction in participants in the trial from seven groups totalling 105 students to three groups totalling 45 students arose partly from the difficulties that students and staff experienced in organizing access to the equipment at times of maximum demand. In addition, part-time staff were unable to continue with arrangements which took them beyond their contractual commitment to the Institute.

We found that without technical assistance being continually available, simple mechanical difficulties, (like running the wrong video disc for a chosen unit of the programme so that video and text failed to synchronize, or failing to notice loose leads) were a common source of embarrassment for Humanities staff who were not familiar with electronic equipment. This perhaps contributed to the decision of some staff to withdraw from the trial. While generous technical support was available to the project, it is clear that staff intending to use such a system and material must become personally comfortable with the technique by having ample non-threatening opportunity to practise using the equipment before incorporating it into their classes.

Student reaction to the use of interactive video

Questionnaires were distributed to all participants in the pilot scheme and completed by those of the students who were still accessible to the project team in April/May during their next unit of work. The response rate can be seen from the following table:

Course of study	Number of students	Number of respondents
Building Engineering	17	11
Computer Technology	28	21
Total students still accessible to trial	45	32

Table 1: Accessible respondents in "Writing For Results" survey.

A copy of the questionnaire is appended to this paper. In the questionnaire, we asked the students to estimate how long it took them to work through the modules, how they had liked working in a small group and whether they had reviewed any of the material since finishing Unit One of the Written and Oral Communication Course (questions 1 to 5). We also sought their (forced) opinion of the most convenient location for further use of the equipment and their perception of the chief benefit of the material relative to other styles of instruction. (questions 6 and 7).

Questions 8, 9 and 10 determined the student's previous experience of computing, and question 11 asked them to rate their own performance in English compared to their peers in the course. Finally, questions 12 and 13 were open questions which allowed the student to comment on what they perceived as the best and worst features of the the "Writing For Results" material.

The following comments were received as responses to the open questions 12 and 13. These responses are presented with the reminder that only 45 out of the initial 105 students were accessible to the project, and of these, 32 responses were gathered. Also, we should comment that the respondents were the most consistent members of their class in terms of both attendance and submission of written work.

Some of the advantages seen by students were:

> makes the person using the system more aware ... he pays more attention to video because he is interested.

> Excellent acting and production. Ideas clearly presented ... easy to understand.

> It was a unique and I found effective way of learning.

On the whole students responded positively to the interactive video even though it took them longer than an hour to work through each module and some had some misgivings about the small-group style of study:

> Not enough equipment (video, keyboard) so we have to share with many other students and never touch the keyboard.

Others found the material boring:

> After the first disc the rest became rather repetitive in the film and the questions asked.

Most rated the material highly. Two thirds of the Computer Technology students said the material compared most favourably with the conventional instruction, agreeing that "This material gives a realistic picture of life in a business firm which I wouldn't otherwise have." (Q. 7(a))

There was a strong preference among students for future location of the material in the audio-visual section of the library. This probably relates to the limited hours of opening of the Humanities and Educational Development Department venues relative to difficulties with the students' timetable referred to above. For example during the first semester the EDD was shut at lunch time and after 4 pm, except by special arrangement. Despite this more than 50 students used the interactive video while it was housed in EDD in 1987.

Some students and staff noted the relative inflexibility of the IIS programme, especially in the first module which restricted the exit/review possibilities. Other comments were :

> No scoring incorporated
> No questions can be asked. Need a general scoring result at the end of the module.
> Doesn't have a printer to print out a permanent copy of questions and their answers.

It was common for students to have difficulty catching what they considered to be key details, for example the names of persons on the screen, although this might easily be addressed by offering users a scenario introduction rather than the topic summary which is currently supplied.

Preliminary analysis of student performance.

It is the subjective opinion of the project team that as a consequence of using Interactive Video, many weaker students made significant gains in language competence. Their grasp of the context in which formal written communication takes place and of conceptual material crucial to consciously increased mastery over written language appeared to show considerable improvement.

In common with the staff in the Educational Development Department who use of this teaching and learning method in parallel Language Development work, we suggest that the introduction of interactive video into language programmes provides an important alternative learning focus for the student. This encourages student-centred rather than teacher-centred learning, a central component of the culture of tertiary education. In addition, such material appears to consolidate what students have already learned, and when used by a small group, provide opportunity for useful conferencing and peer interaction. As indicated earlier, we should advised some caution in the use of this technique for groups of students, particularly at the lower level of course offering with relatively unsophisticated students.

Interactive video was specifically designed for individual response, but notwithstanding the advantages increased utilization and the benefits of peer interaction, the forcing of the technique to service several students simultaneously needs increased teacher involvement and an added dimension of sensitivity by the teacher to the dynamics of group interaction.

The final point to be made in assessing gains by students in this trial is to note that for many this was an empowering exposure to new technology. A significant subgroup said they had no previous experience in computing (4 of the 21 students in Computer Technology and 4 of the 11 second year Building Engineering students) but were nevertheless able to complete a significant proportion of the course. As with language skills, we believe that computing skills are more adequately learned in a meaningful context, and the long-term benefits of this type of exposure to computer technology cannot be overlooked.

Conclusion.

The results of integrating interactive video material into the "Written and Oral Communications" course at the Footscray Institute of Technology suggests that this method offers a useful complement to conventional instruction techniques. Furthermore, its use in small group settings in the context of classroom learning offers students the chance to make significant gains in the appreciation of new strategies for formal communication, and in mastery of the principles of formal writing.

Two key sources of difficulty for Institutions seeking to develop the use of interactive video technology as a means of instruction appear to be the provision of suitably supervised access to students, both in terms of location and time, and the necessity to provide effective preparation for those who will supervise the learning process.

Bibliography

Milheim, W. D., and Evans, A. D., (1987). Using Interactive Video for Group Instruction, *Educational Technology*, June, 35-37.

Acknowledgement

The authors wish to acknowledge the assistance of Dr. James Sillitoe, Head of the Educational Development Department at Footscray Institute of Technology, in the preparation of this paper.

Sheila Byard is a lecturer in the Humanities Department at Footscray Institute of Technology and has been teaching tertiary students and other adult learners for over twenty years. Whilst employed in the Adult Migrant Education Programme she was involved in several projects based on self-paced learning principles, and at FIT has been particularly interested in schemes to encourage language development opportunities for disadvantaged students.

Neil Shaw, formerly a senior lecturer with the Physics Department at Footscray Institute of Technology, is now heading a series of initiatives which involve educational technology for the enhancement of learning by disadvantaged students. Dr. Shaw was the recipient of three significant CTEC Equity grants during 1985-7, and has developed as series of interactive video and interactive audio programmes suitable for use in tertiary institutions.

Simulation and queuing theory: A combined approach

W.W. Read and D.A. Waugh
Department of Computer Science
James Cook University

CAL applications in the field of mathematics have not been extensive; however, CAL techniques have been applied at different levels in applied sciences for some time. This paper focuses attention on simulation as an aid to understanding queuing theory. In courses on queuing theory, the purpose of this is two fold - in a purely mathematical course, tutorial time can be more efficiently used to increase students understanding of the simulation approach. In a systems course more time can be spent on theory, with less lecture time spent on the practicalities of simulating queuing systems. A basic model is presented together with the software design and initial implementation on a Macintosh Plus personal computer.

Queuing and queuing systems are fundamental to modern civilization, and almost everyone has had contact with a queue at some point in their lives. Who hasn't been caught in a queue at a store or supermarket, and wondered if the service could be arranged more efficiently? Unfortunately, queues appear to be increasing in frequency (as well as length!) and some knowledge of systems analysis and queuing theory is relevant to an expanding range of degrees and diplomas. This imposes some responsibility on tertiary institutions to provide pertinent courses on the subject, particularly when enrolled students are not majoring in mathematics or systems subjects.

The teaching of queuing theory can be roughly categorized as either a theoretical mathematics approach or a systems approach. In the mathematical approach, attention is usually focused on obtaining queuing system statistics analytically; simulation may be given some attention, in a much more minor role, than the analytical analysis. On the other hand, the systems approach usually incorporates simulations as the main vehicle for obtaining system statistics; theoretical results for some systems may be quoted with little of the mathematical background given. A student enrolling in both types of courses usually receives a good grounding in most aspects of queuing systems.

However, a student wishing to obtain some knowledge of queuing systems with a minimal enrolment requirement may miss out on a large proportion of the knowledge required for even a basic understanding, if he or she completes a course in only one of the above options.

It is the intention of this paper to focus attention on the development of a simple simulation model implemented on a Macintosh Plus personal computer. The model is proposed in order to aid both approaches, so that more lecture time can be spent on simulation theory in the mathematics course, and "pure" queuing theory in the simulation course. In this way, it is hoped a student enrolling in a minimum of courses will receive as broad a grounding as possible in queuing systems.

A Simple Queuing Model

A large variety of simulation models and languages (Melamed & Morris,1985;

Ramakrishnan & Mitra,1982) exist today, and it is not the intention of this work to attempt to develop a completely new model or language. Instead, a simple model will be used to enhance the teaching of some of the fundamentals of queuing simulation, rather than obtaining knowledge about a queuing system as the ultimate goal. Students completing a full course in queuing theory usually become proficient in the use of a model and language. However, time must be spent to gain a reasonable degree of proficiency in using such a system model, time spent learning about the model and language, rather than queuing. A student who is enrolling in a minimum of courses could possibly spend this time more constructively on queuing theory, both pure and simulation. However, the fundamentals of simulation must still be learnt. Computer graphics provide a very efficient teaching medium (Benzon,1985) and with the profusion of such relatively cheap graphics orientated micro-computers as the Macintosh Plus, also a viable one. One method of simulating a queue manually is via a table. The number of people in the queue,current departure times, arrival times etc. are arranged in columns, and updated as the simulation progresses. As noted by previous researchers (Arganbright, 1985) electronic spreadsheets provide an excellent interactive teaching medium for mathematical modelling, and a variation of this approach will be followed in this work. It has been noted (Sonenberg,1985) that computers are not used very extensively in the teaching of mathematics; this is one area in which computers should and can play a major teaching role.

It has been the author's experience that students (and consequently tutors and lecturers!) spend a lot of time, usually in a great deal of confusion, learning the very basics of simulation. An example is in the use of random numbers to simulate points from a given probability distribution - a typical mistake is using random numbers more than once. Another area is in the simulation of service time - quite often students simulate service times when the queue is empty. The knowledge required to overcome mistakes such as these is very basic and a minimum of lecture time should be (and usually is) spent on it. However, a student must learn these points, and the appropriate place is in a tutorial. The ease and simplicity of using a Macintosh Plus, makes it an obvious vehicle to use to teach students about fundamentals such as these.

The basic model will be a G/G/m queue (General inter arrival and service time distributions, m servers). Although, for given inter arrival and service time distributions, this queuing system is easily simulated, there is no analytic solution (Kleinrock, 1975). The simplicity of simulation will allow attention to be focused on the basics of the simulation approach, rather than obtaining queuing statistics from a complex problem. The features the model must incorporate are as follows:

1. Inter Arrival and Service Time Distributions: A selection of analytical distributions must be available, such as negative exponential, poisson, etc. As well, some capacity for user supplied discrete probability distributions in the form of histograms must be provided.

2. Random Number Generation: Initially, random numbers are to be selected individually by the user, with some provision for automatic selection later.

3. Error Detection and Recovery: A comprehensive set of common mistakes and an appropriate response must be catered for. The response must be available at different user selected levels (e.g. beginner, intermediate, advanced), and employ the graphics capacity of the Macintosh Plus as much as possible.

4. The simulation must be presented visually as well as in tabular form.

5. Design should be efficient and as user friendly as possible, with a comprehensive help capability.

Simulation Environment

Initalization - the user is given the choice of re-activating a previous simulation session, or commencing a completely new session. At any point during execution the session may be saved, under a user defined name, or terminated. In either case a summary of the statistics associated with the simulation is presented. The level of help must be specified as either beginner, intermediate or advanced. Beginners will be modelling a G/G/1 queue, whereas intermediate and advanced users will be modelling a G/G/n queue, and therefore must specify the number of servers in the queue. The inter-arrival/departure distributions may be chosen from a range of analytical distributions, or a discrete histogram input. In the latter case, the user is required to supply the appropriate step functions.

Options - a graphic display of the queue during simulation is one of the main options available, with the default being a tabular summary only. Breakpoints can be set, at which times the simulation is stopped, and system statistics displayed. The simulation can also be "stepped" through automatically in constant time increments, with a statistical summary displayed at each step. Percentiles for the various statistics may be set, with the default taken as 5%, giving 95% confidence intervals on average statistics. The default service order is first-in-first-out, with other service orders (e.g. last-in-last-out, random) available. The number of people actually in the queue initially may also be set, with the default taken as zero.

Simulation - with all the parameters and desired options set, simulation of the queueing system can commence. A random number may be selected from a table, and used in conjunction with a graphical representation of the appropriate cumulative distribution function to simulate an inter-arrival/departure time. Alternatively, the inter-arrival/departure can be generated automatically from the appropriate distribution. This is then used to obtain an arrival/departure time, and entered into the worksheet. The worksheet is then updated to the next arrival/departure.

Error detection and help-upon detection of an error, the incorrect entry in the worksheet is highlighted, and the state of the queue at the time of the error displayed. For instance, a departure time entered into a server's column when no-one is in the system first causes the incorrect entry to be highlighted. The visual display of the queue then shows a dotted outline of a person being served, indicating a nonexistent service time. A very brief error message is displayed, with further help an option. The three help options range from beginner to advanced, with the higher levels progressively briefer versions of the beginner level advice.

Software Design

The basic system is composed of the following meta-modules, (see fig 1), I/O Controller, Queue Simulator, and the File system. The user interface is designed to make maximum use of the toolbox routines provided by the Macintosh operating system. All input and output is directed through a window of some form, either a dynamic display or a dialogue box. A facility is provided to save a session in progress for resumption at a later time.

The Queue Simulator is designed with the following sub-modules, (see fig. 2), Command Processor, Simulation Controller, Queue Editor, and the Model Generator. The Command Processor handles the interaction with the user through the I/O Con-

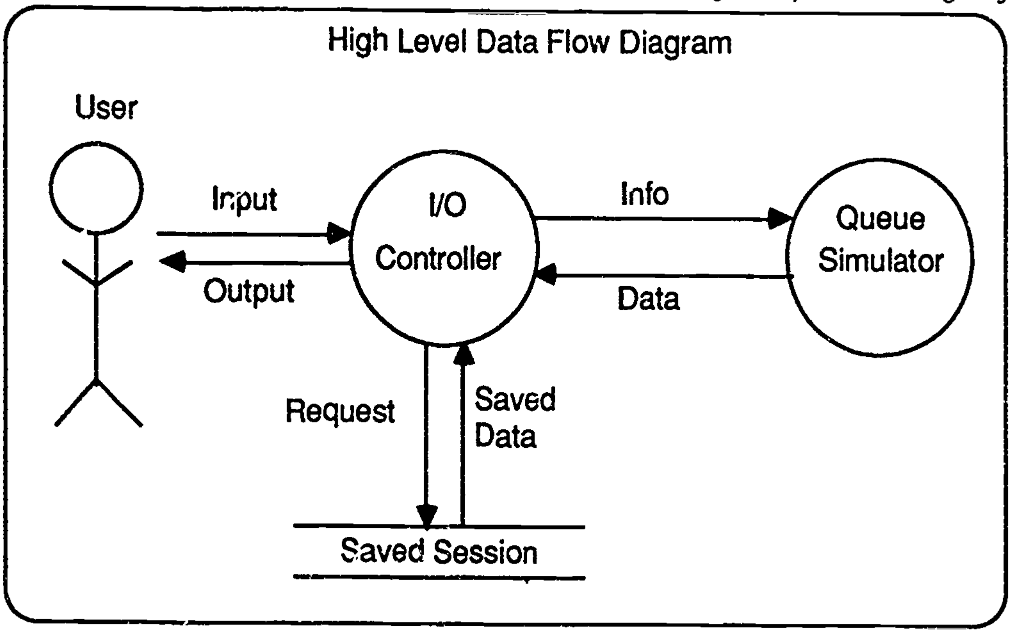


Figure 1

troller, processing the various commands that a user may select from the various menus. Editing information is passed on to the Editor module while commands that concern the running of a simulation are passed on to the Simulation Controller. The Queue Editor module controls the user's input to the system including the entry of simulation data and completion of the details at each discrete step of the queue being modeled. The Queue Editor gets the required results for a specific distribution by extracting the appropriate formula from a stored library of distribution formulas. If parameters are required by the user, a request is made for a valid value for each parameter involved. The Model Generator actually controls and models the present and previous states of the queue. The details concerning the queue's status are kept in the Queue Status data sink/source. The Simulation Controller handles the actual simulation of a queue by the system for demonstration purposes. Commands such as setting breakpoints and stepping through the simulation are handled by the Simulation Controller. The current status of the displays and any necessary modifications are eventually passed on to the I/O Controller through the Command Processor.

The machine specific details are restricted to the I/O Controller module so that porting to another machine will limit the scope of the modifications to this module.

Summary and Conclusions

A basic queuing model has been presented, together with the software design and preliminary implementation on a Macintosh Plus personal computer. The model is aimed at assisting the learning of the fundamentals of simulation techniques, particularly when the student is enrolled in a minimum of queuing courses. In view of this, the focus of the design is in the basics of complicated systems. The Macintosh Plus has a comprehensive graphics capability, and the user friendly features have been exploited to provide an easy-to-use environment for the novice computer user. The simulation model is available at a variety of levels, so that the more sophisticated user can avoid some of the details so necessary to the novice.

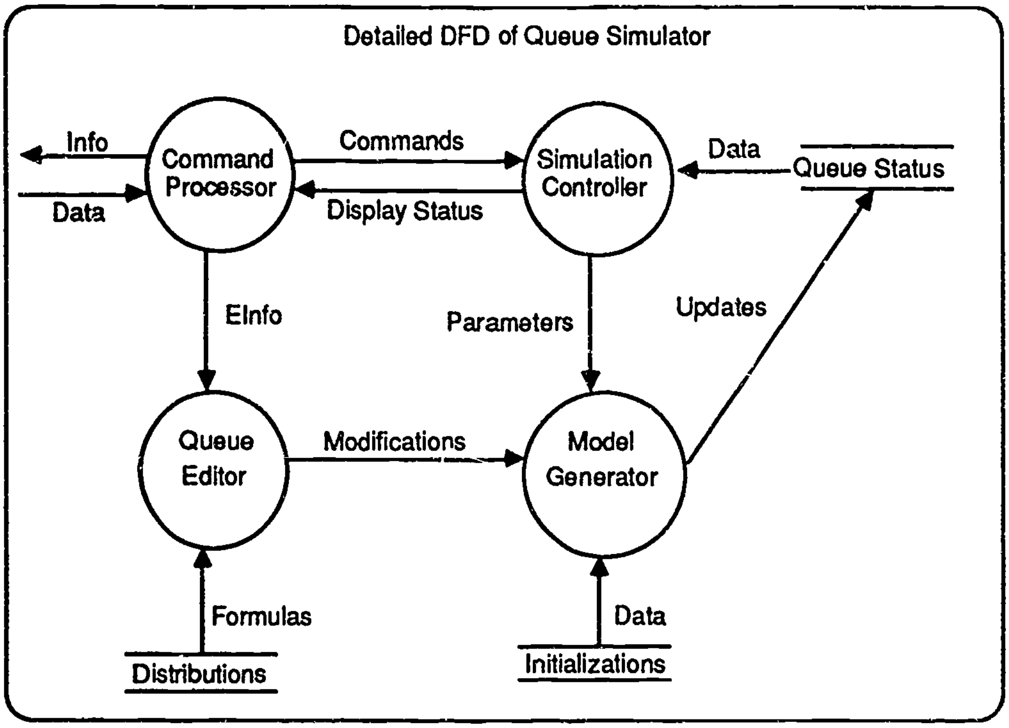


Figure 2

The software is at present in the preliminary stages of development, and completion and comprehensive testing with students is required before any detailed conclusions can be drawn. However, the authors feel that the model will provide a valuable learning tool, particularly when used in conjunction with a theoretical course in queuing theory. The model appears to be an ideal vehicle for applying CAL techniques in the field of mathematics, and so may play a non-trivial role in this field.

References

Arganbright, D. (1985). *Mathematics and Mathematical Modelling Using Spreadsheets*, Proc. Conf. CALITE, Melbourne, pp. 141-150.

Benzon, B. (1985). *The Visual Mind and The Macintosh*, Byte, pp.113-130.

Kleinrock, L. (1975). *Queuing Systems Volume 1: Theory*, Wiley-Interscience.

Melamed, B. and Morris, R. J. T. (1985) *Visual Simulation: The Performance Analysis Workstation*, IEEE COMPUTER, August, pp. 87-94.

Ramakrishnan, K. G. and Mitra, D. (1982) *An Overview of PANACEA: A Software Package for Analyzing Markovian Queuing Networks*, Bell System Technical Journal, Vol. 61, No. 10, pp. 2849-2872.

Sonenberg, E. (1985). *Possible Impacts of Computers on the Teaching of Mathematics*, Proc. Conf. CALITE, Melbourne, pp. 133-140.

Dr. Read was born in Ayr, Queensland. He received a Bachelor of Science with First Class Honours in mathematics from James Cook University of North Queensland in 1982 and subsequently completed a PhD at James Cook in 1986. During the course of his doctoral work, he was employed by the Civil and Systems engineering department as a Research Officer. He is currently a lecturer in the department of Computer Science at James Cook, where he lectures in algorithms and data structures. His research interests include sorting, signal processing, computational mathematics and computer aided learning.

Instructional systems design: The hidden agenda

Rod Sims and Pascal Grant
Computer-Based Training Consultants
Expert Solutions Australia

Often touted as the messiah of cost-effective training, Computer-Based Training (CBT) has only recently begun to deliver the goods. Trainers who once believed novice authors using simplistic authoring systems were capable of developing high-quality courseware, all at minimum cost, now recognize the need for experience and quality. Current emphasis is now placed on the need for instructional design and experienced authors for effective courseware applications. This paper quantifies the real costs of developing a CBT solution from the Instructional Systems Design (ISD) to the actual authoring of courseware. Specifically, the time and costs involved with ISD and the variations in courseware format are identified as major determinants of the overall costs of CBT applications. Through defining a "check-list" for courseware development, the authors provide a guide for estimating the effort (time, materials and cost) required for producing computer- delivered training. From a practical and experiential basis, the conclusions of this paper are that CBT has and will continue to provide a significant cost-saving potential for training.

There is little doubt that microcomputer based quality Computer-Based Training (CBT) courseware is an extremely powerful training medium. This is especially true if an organization wishes to train large numbers of individuals, spread over multiple sites and the subject matter is not likely to be volatile.

The issue therefore, is not whether CBT can survive as an effective complement to existing forms of training, it already is quite successfully in many Australian organizations. Rather, the central issue concerns the Quality aspect of CBT courseware. The essential criteria for a successful CBT strategy is not whether one has developed hours of CBT slowly replacing traditional forms of training and seemingly saving the organization precious dollars. What will determine the effectiveness of any CBT project is whether the training addresses the learning requirements of individuals in the field.

CBT offers a unique characteristic compared to other forms of training, it is interactive. The learner can be tested in a private environment without intimidation, he/she can practice until mastery is achieved and with the use of sophisticated graphics and animations, can be presented with highly effective learning strategies. If organizations fail to meet the criteria of learning effectiveness, given the powerful characteristics of sophisticated interactions - CBT can and will be relegated to the bottom of the training cabinet.

The purpose of this paper is to draw attention to the issue of creating quality courseware. In order to achieve quality courseware, serious attention needs to be devoted to the following aspects:

1. Instructional Systems Design (ISD)
2. Authoring
3. Project Management
4. Project Personnel

Instructional Systems Design

In undertaking any CBT project many organizations have failed to take into account that before any CBT development can take place, an extensive analysis and design component is essential. Indeed the predevelopment stage of creating quality courseware referred to as ISD can often take more effort and time than the actual authoring stage. As an estimate of the time taken to develop an hour of CBT, courseware development groups consistently devote up to 60% of the total project time in the ISD component.

Instructional Systems Development is not a new methodology, it grew out of U.S. Defence training projects in the early 1960's (O'Neil, 1979). Since that time it has grown to become a highly successful and rigorous means by which training needs have been addressed for a multitude of applications. The distinction between design and development is not discussed in this paper, as the authors see the terms as synonymous (Romiszowski, 1986).

ISD has emerged in response to the pressing need for a more effective, efficient and reliable educational system. It involves aspects such as task analysis, behavioural objectives, criterion reference testing, individualized instruction, formative evaluation, and media selection. While ISD was originally developed in the context of military training, it's use has spread to all domains of instruction and is probably the dominant instructional methodology existing today.

ISD in the CBT development cycle is synonymous to the systems analysis phase of any D.P. undertaking. The methodology serves four basic purposes:

a) Improving learning and instruction by means of the problem solving and feedback characteristic of the systematic approach.

b) Improving management of instructional design and development by means of the monitoring and control functions of the systematic approach.

c) Improving evaluation processes by means of the designated components and sequence of learning events, including the feedback and revision events, inherent in models of systematic instructional design.

d) Testing or building learning or instructional theory by means of theory-based design within a model of systematic design.

The purpose of a systematic approach to instructional design is that "it encourages the setting of a design objective, and it provides a way to know when that objective has been met" (Gagné & Briggs, 1974, p. 228).

ISD is a proven and tested method of ensuring effective training for any CBT undertaking. CBT can only be effective if the design stage has been systematically completed. ISD offers the opportunity to ensure that all the various components of the design process have been completed and validated in order to maximize learning effectiveness. ISD will provide a detailed list of learning points, suggested learning pathways and required interactivity. The subject matter and training objectives are reassessed and the lesson is storyboarded in full before any authoring can commence.

Turning to the costs consideration of Computer-Based Training development, the instructional design aspect becomes not only an essential element for learning effectiveness, but will also ultimately determine the resource costs of the project in terms of time taken. In costing CBT projects, there are certain variables ensuing from the ISD stage that will ultimately affect costs and resources. These include:

- the amount and complexity of the subject matter
- entry level of students
- availability of subject matter expertise and documentation
- "system" accessibility for training development
- the productivity of the authoring system
- the instructional strategies to be employed, i.e. simulation, tutorial, game or drill & practice
- the complexity of the Computer Managed Instruction, whether basic student management or full database integration.

It is not the intention of this paper to expand further on these cost variables. However, it should be noted that the design considerations need to incorporate a detailed specification of the training requirements.

A word of caution should be noted concerning costs of quality courseware development. If a designer wishes to employ complex instructional strategies with multiple branching and implicit student modelling as the determinant of progress through the lesson, development times will ultimately increase in proportion to the level of complexity.

This raises an important question concerning the differing levels of quality courseware currently being used in industry. If business, government and industrial groups recognize the costs of training as the most important CBT selection criteria, it will come as no surprise that CBT courseware developers need to design and develop courses cost effectively. Consequently, a compromise must be made between instructional effectiveness and CBT excellence (Hitchcock, 1987).

Constraints on development times imposed by commercial realities may indeed limit the design process considerably. This is exemplified by the low level of business activity with Intelligent Tutoring Systems, perhaps the most advanced form of CBT. One of the basic impediments is that one hour of intelligent instruction delivered by computer may take up to one year to develop, as compared to the 150-200 hours for the development of one hour of "traditional" CBT using pre-determined paths and an expository learning methodology. The basic steps of ISD are shown in Appendix 1.

To conclude this section on ISD, any organization contemplating a CBT project should ensure that the required ISD becomes a major component of the project. Perhaps more importantly, the designer must fully understand the cost implications of determining the detailed project specifications.

Authoring

In the field of Computer-Based Training (CBT), authoring is frequently given top-billing whenever courseware and authors are discussed. More often than not however, the notions of CBT authoring are based on misinformed or simply incorrect statements describing the facilities of a particular authoring system or language. This section provides a realistic definition of authoring by exploding certain myths surrounding authoring and detailing the relevant issues for commercial development of CBT.

Authoring is the process of creating computer-based displays which are presented in an instructional sequence according to specified criteria.

To be able to author, a number of prerequisite steps must have been completed. First, an instructional event (curriculum, course or lesson) must be identified, with the assumption that CBT has been validated as the most appropriate delivery media.

Second, this event will have been subjected to Instructional Systems Development processes to provide a set of sequenced instructional tasks, objective(s) for each task, (optional) assessment protocol for each objective, evaluation procedures and specification of macro and micro instructional strategies for the courseware itself.

Third, a subject matter expert in conjunction with an experienced author produce "storyboards" which specify the content material and interactions, linked to objectives, which are to be displayed to the student. The storyboard also defines specifications for branching and review.

Having completed these steps, authoring can begin, using whatever authoring tool the organization has installed. With the wide range of authoring tools available, the type of courseware developed is highly dependent on the facilities and options provided by the authoring software, and the capability of the author to stretch the system beyond its documented capabilities.

For example, the WICAT "WISE" authoring system allows the author to generate a complement mode whereby graphics may be overlaid and the colours of the two graphics mixed to create a third colour. However, experienced authors have discovered that this mode, when combined with another mode, can allow the overlaying graphic to be erased from the screen without affecting the original graphic. Using this technique can result in exceptional student displays. Such use of this technique only comes after many months of experience with the authoring system.

Authoring tools come in two types: authoring languages and authoring systems. The most commonly used authoring languages are Control Data's TUTOR, PILOT and BASIC. For authoring systems, the choice often seems wider than the prolific spreadsheet and wordprocessing packages!

Representative of current authoring systems are WICAT's WISE, Control Data's PCD3, MPT's PC Class, Compaq's Top Class, Microcraft's AUTHOR, Progeni's FORGE, Mcdonnell Douglas's AIS-II, AIP's COURSEMASTER, MARCONI's MANDARIN and Goal Systems' PHOENIX.

Almost without exception, these products are marketed as being "easy to use" and requiring no "programming" skills.

However, as more and more of these systems have been made available to the writers for evaluation, it is apparent that there are two realities.

The first reality is that all the systems allow authors to create graphics and text using similar WHAT YOU SEE IS WHAT YOU GET (WYSIWYG) editing procedures, and that the resulting displays can be linked into a simple sequence or branching structure.

The second reality is that to achieve interactive learning and full use of the media, the author will have to resort to programming concepts, if not actual programming!

For example, one of the authoring systems evaluated by the writers allows the author to *create* a sequence of graphics using WYSIWYG editing, but then requires the author to text-edit an authoring language if any changes to the original display are required. In addition, any controls required within the delivery system must be programmed in the authoring language. In direct contrast with authoring systems which allow the author to manipulate the display objects on screen, this second reality highlights the the misinformation which is published concerning "ease of use" and no requirement fro programming skills.

The most appropriate method for differentiating authoring systems is to examine the design philosophy behind the system - and of course its year of release. In some instances, authoring systems have been designed to take minimal use of a microcomputers options, with template screen displays and limited graphic options. Conversely, other systems have adopted an approach towards expert systems, with the facility to automate some of the instructional design logic. Progeni's FORGE and Control Data's PCD3 are typical of this genre.

Exploding the Myths

There are two authoring myths:

a. it is easy
b. novice authors can create high-quality courseware

Authoring is *not* easy. Certainly it is far removed from the complexities and structures of programming languages, but it is not easy. A typical authoring system will take 4-6 months to master. Mastery implies the author's ability to contain the full options of the authoring system with little if any use of manuals or help facilities. Experienced authors can however adapt to other authoring systems more rapidly. The writers experience has been that a "good" author can convert a storyboard into computerized instructional displays at the rate of approximately five per hour, which This varies according to the type of lesson and the authoring system used.

It is true that the commands used to create display elements or objects are easy to learn, but the authoring of instructional material is a demanding task.

With respect to the second myth, claims have been made which suggest that novice authors have created high-quality courseware in a matter of hours (Sims, 1986). These claims are simply false, as authoring systems take time to master and high-quality courseware implies interaction. Both factors combined mean that effort is required to produce such courseware. For example, a substantial courseware module will contain at least one hour of instructional material. To create one hour of courseware takes from 175 to 200 hours. This translates to some **three hours per minute.**

What then can novice authors create in a matter of hours? Some interesting displays perhaps, but certainly not high-quality courseware.

The preceding comments have shown authoring to be a skilled task and a significant component of the ISD process. The following discussion analyses authoring in terms of the commercial development of comprehensive, high-quality courseware packages.

After the ISD and storyboarding, the author must decide on the various courseware attributes. For example, courseware standards are essential for the consistency and continuity of lessons. Currently, these standards include a screen border with corporate logo and module identification on the top line and student control prompts on the bottom line. Similarly, standards must be defined for text colour, highlights etc.

One of the most important standards to define is that for interaction, which is the essential element of successful high- quality courseware. Questions relating to the number of attempts, automatic exiting and feedback must be resolved. For example, does the student continue to make a response to a question until correct or will the judging process terminate after a specified number of attempts or condition has been reached.

The responsibility for interaction lies in the storyboarding activity - it should not be based on a set number of text screens followed by a question, but rather on asking the learner to interact by using knowledge already gained. Such an approach also signifies a move away from restrictive frame-based systems to more substantial, flexible and effective systems.

When authors are familiar with the full capabilities of an authoring system, the interaction of the courseware will become the major component of the instruction, thereby increasing the dynamics of the learning experience.

With storyboard in hand, the author can commence translating static displays to the dynamic display screen. One of the major flaws with many courseware packages has been the failure to use the graphical capabilities of the available technology. This suggests that the ISD team were not conversant with the real capabilities of CBT. It has always been bewildering why screens capable of highly interactive and dynamic learning events have been relegated to electronic page-turning.

The success of CBT depends on the effectiveness of the courseware. Courseware is a learning product and needs to be interactive; it is not an information source which, if necessary, can be page-turning.

Authoring Costs

The cost of authoring depends on what the courseware entails; as each additional feature is added the development time and costs increase. These features are often dependent on the facilities provided by the authoring system, and as mentioned earlier, it is cheaper and more simplistic systems which provide fewer options. As such, they can promote the quick development of courseware, but at a sacrifice for full use of the display screen. However, it is also important to keep in mind that over indulgence in courseware development should be minimized (Hitchcock, 1987).

As each option is added, the development effort (ISD and storyboarding) and authoring effort require more time and therefore, in a commercial environment, incur more costs. The authoring options below are placed sequentially in terms of ease of transfer from storyboard to display:

a. Text: Normal size
b. Text: Varied size
c. Text: Alternate character set
c. Text: Coloured
d. Graphics: Circles, boxes, lines
e. Graphics: Coloured
f. Graphics: Rotated, sized
g. Response Judging: Multiple Choice, True/False
h. Response Judging: Short Answer (to 256 characters)
i. Response Judging: Student Interface Devices
j. Feedback: Text
k. Feedback: Text, graphics
l Feedback: Additional response judging
m. Branching: Linear
n. Branching: Student Controlled
o. Branching: Author Controlled
p. Animation
q. System Variables to control display/branching logic
r. Data Collection for student records/management
s. Student Interface Devices: Touch, light pen, mouse
t. Audio and Video

Authoring: The future

With the continual development of authoring systems and the increase in microcomputer capabilities, the future of authoring is bright. What remains is for the developers to incorporate all the useful technologies such as CAD/CAM, A.I. and Expert Systems with current authoring systems to provide trainers with software that will assist in more efficient generation of effective training.

In the long run however, authoring systems by themselves are non- productive. The important issue is to find the authors who can use them efficiently and produce training materials which are effective. With CBT courseware on hand the trainer has a significant training option.

Project Management

The role of project management is a crucial aspect, especially when the number of hou..s to develop one hour of interactive CBT is in the order of 170-200 hours. For the same reasons a D.P. project would be managed from start to finish, CBT should be as carefully monitored.

The writers have developed a checklist of CBT project management guidelines in order to facilitate the completion of courseware within specified guidelines. The experience has been that one hour of CBT from initial analysis to the final validation and field testing of the course can be achieved within 25-27 days. During the project a systematic validation is undertaken with the client to ensure that subject matter is correct and instructional strategies suit the intended audience.

Project personnel

Project personnel for CBT development is perhaps one of the most difficult aspects to deal with. Unfortunately there is a desperate shortage of skilled CBT personnel in Australia. The proof of this is that courseware development organizations are constantly looking for experienced ISD specialists and individuals that have had experience in several authoring systems. They are proving difficult to find.

Notwithstanding the present dearth of experienced people, the writers believe there are several essential ingredients to successful courseware developers. These include: computer awareness, training sensitivity, logical and conceptual abilities, customer relations, diligence and unlimited energies in order to make deadlines.

Conclusions

Instructional Systems Development and a mastery of the authoring system are essential for high-quality courseware. Such courseware is not just a product of a fully-optioned authoring system, but the result of careful planning in the analysis, design, development, implementation and evaluation of the training material.

With this approach, effective and relevant courseware is guaranteed.

References

Gagné, R.M. & Briggs, L.J. (1979). *Principals of Instructional Design*. New York: Holt, Rinehart & Winston.

Hitchcock, B. (1987). The CBT Trap. *Data Training*, 4(5), 11.

Kearsley, G. (1986). *Authoring: A Guide to the Design of Instructional Software*. Reading, Mass.: Addison–Wesley.

O'Neill, H.F. (Ed.) (1979). *Issues in Instructional Systems Development*. New York: Academic Press.

Romiszowski, A.B. (1986). *Developing Auto-Instructional Systems*. New York: Kogan Page.

Sims, R. (1986). On the development of high-quality courseware, in G. Bishop & W. van Lint (Eds), *Proceedings of the Fourth Annual CALITE Conference*. Adelaide, December.

Appendix 1: Stages of instructional systems design

Stage 1: Analysis
- Task analysis and selection
- Objectives analysis
- Target Population analysis
- Training and Resource analysis
- Work Model analysis

Stage 2: Design
- Media selection and syllabus content
- Course design
- Prototype and production costing
- Development planning
- Management planning
- Evaluation planning
- Implementation planning

Stage 3: Development
- Lesson scripting
- Lesson entry and supplemental media integration
- Management system production
- Product integration and testing
- Supplemental media scripting
- Supplemental media production

Stage 4: Implementation
- Site preparation and system personnel training
- Initial data collection and testing
- Revision
- Turnover and acceptance

Stage 5: Evaluation
- Formative (pilot study)
- Summative (after 4-6 months)

Rod Sims is an Associate Director with Expert Solutions Australia, specializing in Instructional Systems Design and Courseware Development. Rod has been working in the Computer-Based Training field since 1979, and has worked as a lecturer and CBT Consultant in government, academic and business sectors. With a Masters degree specializing in Computer-Based Training, Rod's current research interests concern the use of authoring systems and the quality of courseware. He is currently managing the authoring of courseware for government departments and manufacturing organizations.

Pascal Grant is an Associate Director with Expert Solutions Australia, specializing in Instructional Systems Design and Courseware Development. Pascal began his career in teaching, moving to Computer Education and Computer-Based Training in 1981. Pascal has a Masters degree in Education, specializing in Instructional Design. His major research interests concern the implications of Intelligent Instructional Design Systems for the development of training materials. He is currently managing the design and development of CBT implementations within several government and private companies.

The use of CAL in remedial mathematics for economics students

Anne Arnold
Department of Economics
University of Adelaide

For several years,the Department of Economics has been conducting a remedial mathematics course for first year students with little mathematical background. Recently,CAL programs were introduced into this course. This paper describes how this was done and considers the success of the project.In particular, the reactions of the students, the benefits to the teaching staff and the many spillover effects on other students in the Department are explored. The performance of the program will be judged by both a survey of students as well as consideration of their performance.

Although CAL seems to be gaining more and more in acceptance, it seems to me that there are a few areas which may not have received sufficient attention or thought, especially when preparing a course. These include, firstly, assessment of the benefits and costs of such a learning program and secondly,the importance of targeting the programs to the particular group of students concerned.I was also concerned to make sure the students had sufficient backup and personal contact with a lecturer/tutor so that they didn't fall behind or have unsolved problems, without losing the advantages of CAL by requiring too much input from staff. This paper is,then, concerned not so much with "how to" conduct a lesson/course or CAL program but, rather, how to assess the program, and a discussion of factors to consider.

CAL has been adopted only to a limited extent in the Department of Economics at the University of Adelaide.Glenys Bishop first introduced us to CAL and while she was in the Department she set up several teaching programs and exercise sets. Unfortunately, little has been done since then, although we still use her programs.

One of the subjects which seems well suited for a high CAL content is a first year course in remedial mathematics for economics students.Mathematics is becoming more and more important in the study and teaching of economics. Students with a reasonably good high school mathematics background are reasonably well catered for with the option of doing mathematics in the Mathematics Department or in the Economics Department; students with an average high school mathematics background generally have sufficient knowledge to cope with the mathematics involved in the core Economics subjects and no inclination to do any more,anyway; but students with very little high school mathematics were in a quandary.While they often recognized their lack of skills and realized how much it hampered their study of economics, there was little they could do. This first year remedial course fills that gap. It is only open to students with very weak mathematics backgrounds and who are concurrently studying or have already passed the first year core economics course. These are usually good students who are highly motivated and will work hard and learn quickly.(The less motivated student will often not take up this option)

I have dwelled on the nature of this subject ar some length so that I can highlight the particular problems in teaching this course and the reasons why CAL is potentially so successful here. First, the students do have very variable backgrounds - while they have had no formal mathematics education in the latter part of high school, their competence ranges from none at all to some familiarity with numbers and graphs etc. Most of them share a deep seated fear of mathematics and high anxiety over the subject. Second,they learn at vastly different speeds. Some seem to pick up the concepts very quickly whilst others need much more reassurance, practice and time. Third, they all require their confidence to be built up. This often necessitates giving them a lot of practice problems that are marked by their tutors or, less preferably, by providing them with solutions. Fourth, they are highly motivated (as mentioned above) and are prepared to work hard at the subject and often undertake extra work in their own time. Fifth,since not all mathematically illiterate students in the faculty choose to do this course, it is useful to be able to open up the course for any students to sit in on, even if they don't formally enroll. Also of relevance is the fact that we seem unable to find a text book which covers the course in sufficient depth or detail and with sufficient examples and exercises. In addition, there is typically only a small number of students doing this course and it is a labour intensive undertaking.

It is clear that CAL is ideal to meet these requirements - it allows the flexibility for students to go at their own pace, progressing rapidly over the sections they find easy or familiar, going slowly and/or repeating the sections they find difficult. Many exercises can be incorporated into the lessons, which the computer instantly marks and tells them if they were right or wrong, giving hints as to the answer if required. This builds up their confidence and also alerts them as soon as they don't understand a concept. They are able to go back and re-use the lessons as often as they like, and not worry about being a nuisance to staff members,nor appearing silly to their peers or their tutor by always asking question, getting wrong answers and so on. In addition, the lessons are available to everyone in the Faculty - including staff members wishing to give their school age children some extra help!! My aim, in the long run, is to set up all of the mathematics for economists course as CAL modules although for reasons given later, I don't envisage that the whole course would be taught solely by CAL. Progress towards this aim is slow... especially as we are about to alter our computer resources away from a VAX running VMS to PCs. This will necessitate rewriting the existing lessons and for this reason we have not moved quickly on writing new lessons. However we have two new suites about which I wish to talk. The first suite is a set of lessons on MATRICES; the second a set of lessons on LIMITS. The programming for both of these was done by Mr.Tony Kiek, a programmer in the University Computing Centre with a special interest in training and education. We worked closely together - I provided a course outline and detailed course notes, Tony did the programming and any subsequent changes and additions. Each package was set up as a series of about five or six individual lessons, each about an hour long and containing explanations, examples and exercises.I do not intend to give a detailed account of the content or mechanics of the lessons, but will point out that they contain provisions for going back to previous sections,"fast forwarding" for students repeating the lesson and a very clever record keeping system. The records kept are the date and time the lesson was started,the user's name, which lesson they undertook, whether they finished and how many errors they made. The only reason for and use of this record keeping is in assessing the packages - using this information we can determine whether any of the lessons are too long, too short, too difficult, or have any particular problems.I envisage that we will not need to continue this record keeping once we are entirely happy with the programs. In addition, we could monitor students' progress but this may not be entirely fair as the students seem to prefer the anonymity and privacy of the computer lessons, so this is not done systematically. I will be happy to provide more details of these two packages to anyone interested.

Of interest now is the appraisal of the system. The amount of time spent writing these two packages was very considerable, although the second was much easier and quicker once we had the experience of the first to draw on. For a small class such as this the costs of involved in tying up the terminals etc is negligible - at the times of the year when the students use these two packages, computer resources are reasonably slack and there is no problem in the students finding free terminals. The limits package requires a graphics terminal which is a constraint as many of our terminals are not graphics terminals, but this did not present a major problem. These are the costs. What of the benefits? Benefits are of three sorts - to the students of this particular subject, to the staff and the wider spill-over effects.

Benefits to the student

A survey was conducted of the students to find out how they felt about CAL lessons in general and the LIMITS course in particular. Unfortunately, because this is only a small class, the number of respondents was low (15) even though the response rate was high (over 75%), and so meaningful statistical analysis is impossible. One of the results that startled me was that before they started on the course, very few of the students did not like the idea of learning from a computer, even though few had used a computer before. One or two were scared that they would not be able to cope, but the majority were excited by the challenge of doing some thing different. This of course put them on the right track for positive learning. After the CAL lessons, all were quite happy with the idea of learning from the computer although some indicated that they would not like to do all their work this way. There were a few particular problems with these lessons that caused some dissatisfaction - namely that two of the lessons were too long. This is very important - although we had test run the exercises many times, we did not allow for the amount of time the students spent copying from the screen. This can be rectified by altering the lessons, giving the students a more accurate impression of how long each lesson takes and warning them to be selective in what they write down. Many commented that the ability to take very good notes was an advantage of CAL lessons but it also took a lot of time. Specific questions were asked about the advantages and disadvantages of CAL, as well as these particular lessons.It seems students see CAL as a useful adjunct to traditional teaching but do not want to lose all contact with lecturers and tutors. Five (out of fifteen) thought that CAL was too impersonal whereas the others said it was not. However all but one thought that the ability to work at their own pace was a very real advantage.In fact, this was probably the most important advantage to the students - they could work at there own speed, pause when they wanted to, go back over some of the sections and repeat whole lessons at a later date. Several students also liked the active nature of this learning - being forced to think about problems on the spot, not being afraid to get the wrong answer,not tending to "drift off as you do in a lecture" and even appreciating the challenge of having a readily identifiable task. In other words, they had something they would start, do and finish, and as with an assignment,they derived satisfaction of completing such a defined task.

The disadvantages that the students cited were on two levels - firstly aimed at the particular package and its length and secondly aimed at CAL. These perceived disadvantages are important and must be considered carefully to ensure students continue to be motivated by and learn from CAL lessons. The major disadvantages students noted were; they missed the verbal explanation of, and interaction with a tutor/ lecturer, they found they lacked keyboard skills and/ or knowledge of computing so they often wasted time. they had to get used to reading, writing, typing and thinking at the same time. they were aware that it is easy to put off doing the session and then to find yourself behind, some said they often wanted to ask the computer questions such as "why"? or "what if"?

Most of the disadvantages that the students mentioned can be easily solved by providing them with more help and instructions before they start and ensuring that they are able to ask questions of the tutors and lecturers. None of the students perceived the problem of trying to write down too much detail from the screens but my feeling is that they did this - given the opportunity, they tried to rewrite the entire lesson. Other problems that I felt arose were that they need to be very honest with themselves and be prepared to use the lessons intelligently.We had one student who raced through the lesson in about ten minutes but made 117 errors!!!

Benefits to the staff

In their survey responses the students indicated that they would not have liked a tutor in the terminal room with them while they did their lessons - only 4 of the 15 would have liked this. They did however like to know that help was nearby - the supervisor of the economics computing rooms is close and my office is opposite the room they mainly used. This means that tutors and lecturers realize considerable time savings as we did not need to be there when the students were doing the work. As these lessons replaced all lectures and most of the tutorials in this topic, we were saved these contact hours. There was obviously less preparation and marking time,too.The students wanted some personal contact with their tutors so we had one tutorial on the topic, but I suspect that this was not really necessary but more due to habit. Against this time saving I did have several students call in to see me and seek help - certainly more than I would during a lecture course but, they were told that this was a new suite of lessons and that their comments were welcome, and many of their queries and problems caused us to alter and improve the wording of the lessons so that these problems should not arise in future years. On the basis of 20 students and five lessons, I estimate the academic staff teaching the subject saved about 7 hours of contact time and 14 hours of preparation and marking.

As well as the direct time saving,there were other benefits to the staff - enjoyment, challenge, learning new skills and so on.These are largely non-measurable but not insignificant.

Spillover benefits

There are many spillover benefit, although many of these may not be apparent initially.Certainly, the availability of these lessons to an audience wider than just the students they were written for is an important factor. I ran a remedial mathematics course for Honours Economics at the start of this academic year and was able to point potential Honours students in the direction of these packages. The advantage to them in particular is privacy and flexibility in the timing of these CAL lessons.In addition just the meagre computing knowledge that the students pick up from this exercise (and it it is no more than being able to log on, log off, and type a few characters) makes them feel that computers are not all that difficult to cope with, and probably helps them when they meet computing in other subjects.

The effects on student learning

Apart from whether students actually enjoy CAL or not and whether it eases the work load of the teaching staff, there is the question of whether it improves students learning and understanding of the topics.With this particular course, there are two aspects to the question;first whether the particular subject helps with their studies in economics in general and second; whether CAL is better than traditional techniques in teaching them the required mathematics. The first question is not as relevant here as the second.With such a small number of students and no way of controlling between years, it is a difficult question to answer.However, my impression is that students who had done the CAL lessons understood the concepts better than students in previous years.Questions in the exam were purposely kept to a standard very much the same as in previous years and the marks were higher in the CAL group than in previous years.However no firm conclusions can be drawn from this as too many other factors varied. It is at least encouraging!In addition,if student results are at all correlated with their enthusiasm and motivation, then CAL should be effective.

Summary

The results of the student survey seem to indicate that students enjoy CAL and will learn from it. In fact only one student was not looking forward to the next suite of lessons and most were more favourable to the idea after having completed one package than before.The particular features of CAL make it especially suitable for a course such as remedial mathematics and the students stated advantages of CAL evidence this. Given that most of the disadvantages cited were either of a technical nature (the layout of the computer rooms, length of lesson and so on) and can easily be fixed,it seems that we should try and incorporate more and more CAL into this subject.There are three limiting factors, however.First, the the students do not want to be left entirely without personal contact and some, although considerably reduced, teaching commitment is still needed. Secondly, the lessons must be technically well presented - what seems to be to the author of the lesson to be a triviality can become very important to the students (for example, whether upper or lower case is required). Thirdly, the time required to set up a well developed, well targeted suite of programs that the students will find interesting,challenging and useful without being too long or too difficult provides a very real constraint.

Anne Arnold is currently Senior Tutor in Quantitative subjects in the Department of Economics, University of Adelaide where she teaches mathematics and statistics, which is where her interest in CAL lies.Her research interests are in many areas of applied economics.

The subject matter of CAL technology: Towards a development methodology

John Barrett
Macquarie University

This paper suggests that: (1) Implementation and acceptance of CAL should be within a clear understanding of the nature and purpose of learning in tertiary educational settings; (2) CAL has a unique and significant potential which has been constrained through a lack of conceptualisation and adherence to inadequate theory; (3) Substantial advances can be gained from adopting a "common code" for representing design, implementation and learning dimensions; (4) Such a coding schema could be developed from cumulative knowledge in the area of instructional design, from CAL research and development knowledge over the past twenty years and from recent developments of the integration of artificial intelligence and CAL; and (5) Theoretical, conceptual and practical considerations for CAL should be founded on the integrative research and debate in the evolving interdisciplinary field of Cognitive Science.

This paper presents some ideas and underlying principles which impinge upon Computer Assisted Learning (CAL) courseware production and its successful implementation in higher levels of learning in formal education and business/industry training.

But, before proceeding to a more detailed discussion of CAL development problems it is necessary to outline some aspects of the role of teaching and purposes of learning in tertiary education into which a CAL system could fit.

Articulating Learning Goals in tertiary education

Given that there is (and seemingly always has been) substantial debate about the functions of tertiary education, the following statements will probably not be acceptable to all readers but will serve as a starting point for discussion about the goals of any tertiary instructional system.

Some of the *teaching* aims of post-school education could be thought of as providing a means for students:

- To gain a meaningful understanding of the structure of the discipline(s) being studied (after Ausubel, Bruner et al.)
- To develop a knowledge base of facts, procedures, systems, techniques related to the subject area
- To form an appreciation of the history of human effort and individual thinking and achievements which contribute to current ideas, research findings, beliefs and values systems in a discipline. This includes intellectual processes, tools, methodologies and outcomes of enquiry and debate which highlight human creativity, frustration, failure, chance interventions, brilliance of reasoning etc.
- To understand the meta-methodologies of the arts, sciences, engineering, the law etc.
- To become motivated or inspired by creative problem solving processes including acquiring a value set for the search for truth, beauty etc

- To develop general skills of criticism, debate, reasoned argument and skills in oral and written communication.

- To develop a repertoire of knowledge/ skill competencies - with a number of particular high level performance skills in specialised areas.

- To fulfil intellectual, aesthetic and social/ moral needs and desires (for each individual) in a socially responsible way and become aware of social and cultural traditions and contexts.

- Develop techniques of inquiry and problem solving ("research" and evaluation skills) and to undertake, manage and complete tasks and assignments by controlling, allocating and applying personal resources and the resources of the local environment.

- To contribute to the development and learning of social interaction skills of one's self and others by interacting through modes of interpersonal and media communication.

This 'goal list' is by no means complete but merely suggests general areas of understanding that learners should gain from a tertiary education program. The list serves to illustrate the points that:

- knowledge frameworks of concepts that are in-filled with facts and procedures and are central to the learning process;

- we are analysing human learning and most (all?) goal statements have an affective component;

- emphasis is on cognitive processes rather than behavioural outcomes;

- an instructional delivery mechanism (human/human or human/machine) has to serve the above purposes *in the first instance*. That is, for example, in an educational setting a delivery mechanism with high entertainment value but low instructional purpose has little place;

- any delivery mechanism must cater for specific needs of instruction in providing a suitable learning environment that can only be truly evaluated in student achievement terms;

- the mechanism for providing instruction must be efficient in terms of the time required for the development process and the effort required for implementation and supervision.

We therefore need a paradigm for instructional research and development which covers educational theory for design and provides a suitable learning/ instruction computational model.

We my conclude that our task is to select a framework for the formulation of a development environment which includes:

- how to capture and represent content

- how to store and deliver content

- how to generate interaction with content and ideas

- how to assess the effectiveness and of the learning process and the efficacy and efficiency of the instructional delivery system

The computer as an instructional delivery device

Here I wish to emphasise the view that "the computer" is not just another piece of educational technology of the same class as an overhead projector. Not only do we have new tools in this symbol manipulating machine, which is an attribute unique to computers, but other aspects such as the speed of retrieval, access to vast amounts of stored information, an ability to bridge geographical and conceptually wide gaps of communication, and the ability to control other devices singling it out as one of mankinds greatest inventions. In addition, the very nature of the computer as a tool for intellectual modelling raises it from a number manipulator to a "cognitive partner" in research, teaching and dissemination in tertiary education.

As various tools and processes evolve the creative powers of teachers in post-school institutions and the army of 'human resource developers' and trainers will invariably demand new software tools for creating educationally more complete (and computationally more demanding) learning environments. There is now appearing a number of high quality authoring tools and languages, including *Course of Action, Course Builder and Guide* and the Apple *Hypercard*. But there remains a number of serious deficiencies which must be addressed if we are to head towards the full capabilities that computers potentially offer.

Deficiencies in Current Approaches .

In previous papers, I referred to the process demands placed upon the developers of computer-based instruction (Barrett, 1983a '83b, '83c, '86a, '86b). These papers analyse design and implementation variables and constraints in courseware authoring. Approaches to CAL implementation generally fall into one, or a mixture, of the following categories and scenarios of usage in tertiary settings:

1 *Application Tools.* Take an application package such as a word processing or statistical package and incorporate it in the classroom with as little adjustment as possible to the curricula.

2. *Problem Solving and Modelling Tools.* Go a step further than (1) above and develop models of the problem solving approaches employing these new cognitive modelling tools.

3. *Packaged CAL.* Use someone else's courseware, commercial or otherwise and readjust the curriculum around whatever is available.

4. *Local CAL development —"The Cottage Industry Model".* Employ whatever skills are available from those (or that person) in the area(s) of instructional techniques, curriculum representation (subject area content), graphics design and coding on computer (programming/ authoring)

5. *CAL —"R & D Centre Model".* Full development tools and resources in a managed organisation. This category is very rare in tertiary education in Australia but a niche of rapid growth in the commercial world.

Deficiencies in any of the above development environments include:

- appropriateness of the specific courseware
- application developers knowledge (or lack of it)
- development tools capabilities
- project management skills and support resources
- financial support, encouragement, recognition for outcomes.

Again I contend (see Barrrett 1983c and 1986b) that significant advances in the widespread use of CAL in Tertiary Education will come about through the development of more appropriate development tools, the building of a cumulative library of an accessible resource base and concerted effort in an efficient CAL production centre and conducted on modern business grounds.

The underlying problem of CAL development

The underlying difficulties facing CAL development and its infiltration into the educational system (beyond financial considerations) is a lack of agreement on the "superstructure" or subject matter of the CAL development enterprise. By this I mean there is no universal agreement of the nature of the instructional enterprise or its particular *underlying coding mechanism*. This position has arisen mainly because:

- learning research was forced into a backwater on the tide of behaviourism. The whole approach has been shown to be devoid of any substantial contribution and should be relegated to its proper place with the "learning" of lower order "reflexive" skills.
- the principles and research findings relating to Adult Learning have, in almost every case, been ignored
- delivery mechanisms have been modeled on the mechanical teaching machines and similar media. As previously stated this has restricted us to a limited metaphor blinding us to what is possible with an entirely different perspective
- there are at least three diverse forces which influence CAL development and the techniques, tools and philosophies from each has marked influence on the finished product -
    1. educational/ learning dimension which operationally encompasses the design of instruction. A vast information base has evolved from research on learning and instruction which should be the basis of any courseware development
    2. the computational dimension which operationally defines the instructional process and evolves a machine environment for the delivery of designed instruction or provides tools for learner controlled explorations
    3. media dimensions — from influences of graphics design, visual media specialists or communication experts.

For too long, many CAL developments have been dominated by the author's our own area of interest or expertise to the exclusion of one or more of the other elements vital to a success courseware production. It is extremely rare that a development environment contains a person or team having the right mix of educational, computing and media/communications expertise.

More importantly, however, development tools have provided little positive guidance for the integration of the distinctive phases of the interpretation of initial ideas through design, layout, coding, testing and evaluation. Most instructional developers, and probably most teachers remain alienated from CAL because of the lack of easy access to either development environments (and tools) or an inability to make changes to courseware that is only "nearly" suitable for their particular classroom application.

In a similar vain, a distinction remains between tools available to designers and tools available to learners in a run time version of the courseware. Learners often have a much more restricted environment during a CAL lesson. Often, developers have commented that they "really got to know the subject area after wrestling with structure and content or 'programming dimensions" but then they only pass on to the learner a constrained path with virtually no "freedom to learn"!

Through computer accessibility and their pervasiveness in all walks of life it is not surprising that a whole host of forces have motivated a diversity of people to develop CAL courseware. Some goals have been more commercial than educational, some processes more computing concerns than instructional goals, and many outcomes amateur rather than professional. They are more often conducted as "back-yard" (read "isolated campus") activities rather than processes of modern production centres.

There is an even greater problem than lack of purpose and lack of co-ordinated research and development effort and this concerns a lack of conceptualisation which is taken up in the section that follows.

Devising a CAL metaphor

Many of man's greatest achievements have been built on a dream - a vision. The use of a suitable metaphor provides not only the visual imagery and emotional stimulus but also provides a concrete framework on which to hang our ideas. Many people know of the chalkboard matrix that gave rise to the Spreedsheet metaphor and the production of Visicalc. Similarly, the "desk top" metaphor has been carried over from Smalltalk to the Apple Macintosh, Commodore Amiga and Atari machines.

More recently a team of "stood down" workers from a Seattle newspaper set out on a market research tour of Washington and Oregon with the general notion of selling the idea of producing advertising copy on microcomputer at newspaper offices. But, in a moment of discussion the four particiants had concurrent insight into the underlying problem they faced. It was not newspapers that had the "need" for collating advertising materials - they already had the process, however backward- but what if individuals had access to such tools? Mr Paul Brainerd, a journalist with a long standing interest in computers, realised the goal was to provide the general populace with full professional publishing facilities. The metaphor of "desk-top publishing" was realised and the idea materialised as PageMaker from the now highly successful Aldus Corporation (Talevich, 1987).

The point here is that we in tertiary education need to be very clear of our goals and purposes to come up with a suitably visioned tool that will help us achieve our goals.

I believe that current CAL is still based in the book metaphor, with a foundation in verbal material and text. Developers and learners need more of a visual base and a suitable metaphor. One suggestion could be a *'space exploration metaphor'*. This alludes to a 'universe of knowledge' which is to be explored and mapped. It includes the idea of various systems (disciplines) and sub-systems (fields) which require exploration and 'documentation' (understanding). In this metaphor the CAL software acts as an electronic guidance system for the journey of exploring new knowledge.

However, there is a far greater and more fundamental deficiency. What we need is a universal code of symbolic representations which will assist us in recording the map of a knowledge universe. And we need this NOT to alter the knowledge structure of the discipline into some phony or forced coding system but to provide the most efficient means of synthesising materials into a formulation for natural learning experiences and its producing the documentation. This can only be achieved through an agreed upon analysis and classification system having its unique nomenclature and instructional/ learning coding system. Our goal then should be to develop a common code for learning environments, with special attention to computer-based learning. The representative code should condense or provide a "shorthand" representation of the knowledge, procedures, skills etc. making up a discipline or sub-disciplinary area. The code should provide a means of cuing the various instructional delivery modes, media and interaction processes.

In search of the Common Code

One could hardly imagine developments in, for example, the arts, engineering or science without the existence if an underlying code - possibly mathematics! Similarly, agreed standard codes for manufacturing design and production are essential. Even the computer industry with numerous "standards" has many common underlying coding schemas. But, something as important as education - specifically instructional design and "learning products" is conspicuously lacking conceptual agreement or functional commonalty in respect of representing the teaching/ learning process. We are in a situation which would be as crazy as each draftsperson producing building designs with no agreed upon symbols or means of expressing specifications. This is even more disastrous when we examine current practice in CAL production.

The overall problem of CAL development becomes clear if we briefly examine the current process of authoring CAL materials. This could be labeled bottom-up, atomistic approach and well entrenched in behavioural methodologies. The 'mechanics' of structuring objectives, formulating modules and sub-sections and the very development process from the parts-to-the-whole encourages a 'dis-integrated' product. More importantly, the "production system" is devoid of any universality in its coding process or its documentation. But it becomes impossible to compare the design, learning processes and possible cognitive outcomes unless there exists some type of common coding system.

It should be possible create a system which is theoretically sound, has universal agreement at the coding level and which has practical attributes of natural ease of use and management. The common code representing instructional content and processes essentially represents the layers of -

- the learner's cognitive processes
- activities and processes in the learning environment
- the intentions and operational instanciations of the author/ instructor

Building a code

The type of coding system suggested in this paper should be built upon three main foundation areas.

First is the accumulated knowledge of instructional design systems technology epitomised by the work of Romizowski (1984, 1985) . Although, as mentioned earlier, much of this work is based on behavioural psychology there is a wealth of synthesised material from systems research which can be balanced by content and procedures from the two areas that follow.

Second, we should take into account the vast amount of information built up from at least twenty years of research and development of computer-based learning systems. These areas of expertise need careful evaluation in order to identify what really works. (See for example the work of Baker, 1978; McCann, 1981; Rushby, 1981, O'Shea, 1983; Steinberg, 1984; Kearsley, 1987; Merrill, 1985 and 1987; Deer, 1987).

Third, the CAL coding taxonomy should by based on a firm conceptual, theoretical and computational footing derived from the evolving discipline of Cognitive Science. This new interdisciplinary field brings together debate, research development from Cognitive Psychology, Linguistics, Computer Science (Artificial Intelligence), Neuroscience and Philosophy. Many of the theoretical and practical problems faced in these disciplines are directly related to such areas as knowledge representation, problem solving and machine configuration and especially human learning that they form the very conceptual rock on which a new science of instruction can begin to built. (See. for example, the writings of Anderson, 1980; Gardner, 1986; Levine. and Rheingold, 1987; .Minsky, 1986).

But what central areas, or design content, should the proposed code encompass? The cognitive elements which must be covered in detail are (at least):

- language - in general and the jargon of a content field or discipline
- perception, attention and visual imagery
- numerical and computational elements
- knowledge structure(s), representation and memory
- cognitive strategies in learning
- reasoning and problem solving strategies and development of expertise
- cognitive development requirements
- psychology of motor performance (as related to competency).

It is readily observed that the above factors are highly weighted in the cognitive domain and suggest a coding mechanism which represents aspects of cognition rather than such 'traditional' dimensions as behavioural objectives, frame sequencing or reinforcement. However, while not excluding affective or cultural influences or interventions in cognitive performance, these are not the primary focus of this approach. The coding schema should first and foremost reflect the cognitive processes taking place in the mind of the learner as he or she interacts with content mediated by the delivery system. In operation the suggested representative code has to serve three masters - representing the content of the discipline, becoming the stored and transmitted code of the CAL delivery system and being a representation of the students learning. Attempts have been made to include these dimensions in modern Intelligent CAL systems but without the benefits a common code would offer.

The ideas expressed here may be considered in terms of a music analogy; the musical notation represents, at various times, the ideas of the composer, sequential and propositional instructions to the musician(s), 'control structures' to conductors, psycho-motor commands to players, music to the audience and sound/ visual/ aesthetic images and experiences to the individual listener - all by means of a workable coding process for recording, storage, retrieval and presentation and reception.

Like music, our instructional ideas and content needs to be represented not only for the idea they 'contain' but how they are to be delivered, ie. on what "instrument" or .audio/ visual mode or media. Similarly, the instructional "concert" must be "orchestrated" and delivered in such a way that makes the whole experience worthwhile in light of the criteria of the composer/ author, conductor/ teacher and listener/ learner.

It should be noted that, even if there are some "intermediate" processes the coding mechanism either holds true to the original authors intentions or, at least, allows a range of variations and interpretations by later conductors and musicians or recording engineers. That is, for example, recording sessions by different cultural groups under different conditions allow local variations while still preserving much of the original ideas. For instruction, provided the original ideas and content are 'classical' they can be preserved and delivered in more modern modes or styles but still convey the original content.

Development processes and the delivery system

The suggested common code approach provides a number of practical advantages to instruction:

- *Accumulated resources and delivery control or access.* The "common code" approach not only provides a common mechanism of understanding the structure and deliver specifications of content over disciplines and across time/ space barriers, it encourages the pooling of resources on a national and international basis. It provides an efficient means of authors identifying or analysing their own, or others, instructional materials. More importantly, classification of vast amounts of material such as that appearing on CD ROM (for example the Microsofts "Bookshelf" reference works - there are now more than 150 CD ROM applications) will prove invaluable to instructional designers and instructors.

- *Templates.* Similarly, templates of accumulated resources on an international basis will advance the implementation and quality of instructional/ learning resources. With the appropriate implementation tools (application 'front ends', CAL shells, designers toolkits, instructional "sub-assemblies" etc) lecturers and instructors should be able to concentrate on the quality of subject-matter content and cognitive processes of the learner - which should be now feasible for any "student model" will be constructed in the same underlying code.

- *A New Authoring Process.* But, given the intricacies of the task and the seeming complexity of any coding schema, how can we possibly hope to author even a ten minute sequence of cognitive learning? The same could be said if we were about to embark on a new thing called music! The answer is virtually opposite to most CAL authoring. The process suggested here involves working from the "big picture", top-down perspective. The author would begin by writing, or capturing the essence of the content as a natural language narrative. The next step is to bring this into instructional code by use of a "Tool for Instructional Editing" (TIE). The author would go through each paragraph by use of the instructional editor marking out deferentially all the "instructional units" and control structures which would be translated into coding symbols like musical notation. It is the wholistic script which holds the key to continuity and integration. If the script is altered, similar to an ideas outliner (such as Think Tank), the text will change place but the underlying structure and representation will remain unaltered but transposed.

The TIE allows for the analysis of content in terms of structure of the content with regard to facts, concepts, principles, rules and propositional organisation, procedural knowledge etc; of instructional variables such as number and type of examples, pacing, questions/ answers, etc; of learner cognitive variables such as memory encoding prompts, consolidation processing activity, reasoning processing etc. The TIE will also be capable of orchestrating delivery modes. This is an "authoring front-end" which assists in the Instructional Design process and, more importantly, "generates" the underlying code in 'cognitive units'.

Overall, the common code approach will encourage the integration of learning through coding which relates to "natural language". Thus, this form of authoring will pave the way, or integrate with, developments in intelligent CAL programs. Its main strength other than practical considerations, is that it is founded on cognition (thought) rather than behavioural manifestations.

References

Anderson, J.R. (1980). *Cognitive Psychology and its implications*. New York: W.H. Freeman & Co.

Baker, F.B. (1978). *Computer Managed Instruction: Theory and Practice*. Englewood Cliffs, N.J.: Educational Technology Publications.

Barrett, J.B. (1983a). *Instructional Demands for Computer Based Learning*. 2nd Australian Computer Education Conference, Melbourne.

Barrett, J.F. (1983b). *Computer-Based Learning: Constraints on Development*. Annual Conference of NSW Computer Education Group.

Barrett, J.F. (1983c). *Designing Computer Based Learning in Tertiary Education - Requirements and Realities*. Computer Assisted Learning in Tertiary Education, Brisbane.

Barrett, J.F. (1986a). *Student Model in Intelligent Computer Assisted Learning*. Proceedings of the 17th Annual Australian Colleges of Advanced Education Computing Conference (p.102).

Barrett, J.F. (1986b). *Towards Specifications for Intelligent CAL*. 1st Australian Artificial Intelligence Congress, Melbourne.

Deer, B.L. (1987). *AI and The Authoring Process*. IEEE Expert, Summer.

Gardner, H. (1985). *The Mind's New Science*. New York: Basic Books.

Kearsley, G.P. (Ed) (1987). *Artificial Intelligence and Instruction - Applications and Methods*. Boston: Addison-Wesley Publishing Company, Inc.

Levine, H. and Rheingold, H. (1987). *The Cognitive Connection*. New York: Prentice-Hall.

Merrill, M.D. (1985). Where Is The Authoring In Authoring Systems? *Journal of Computer–Based Instruction*, 12(4), 90-96.

Merrill, M.D. (1987). *An Expert System for Instructional Design*. IEEE Expert, Summer.

Minsky, M. (1986). *The Society of The Mind*. New York: Simon and Schuster.

McCann, P.H. (1981). Learning strategies and computer based instruction. *Computers and Education*, *5*, 131-140.

O'Shea, T. and Self, J. (1983). *Learning and Teaching with Computers*. Englewood Cliffs, N.J.: Prentice-Hall.

Rushby, N.J. (Ed) (1981). *Selected Readings in Computer-Based Learning*. London: Kogan Page.

Steinberg, E.R. (1984). *Teaching Computers to Teach*. London: Lawrence Erlbaum Associates.

Talevich, T. (1987) *Paul Brainer '70: The Wizard of Aldus*. Old Oregon, University of Oregon, 41.

Software Discussed

Bookshelf Microsoft, Australia.

Course Builder. TeleRobotics International Inc., 8410 Oak Ridge Highway, Knoxville, TN 37931.

Course of Action. Authorware Inc., 8621 Pine Hill Road, Bloomington, Minnesota 55438.

Guide. OWL International Inc., 14218 Northeast 21st Street, Bellevue, WA 98007.

Hypercard. Apple Computer Australia.

John Barrett is a Lecturer in Education at Macquarie University. His research and development interests are in the psychological aspects the use of computers for learning in education and training. He first became involved in computers in psychology and education at the University of Oregon in the early 1970's, graduating with a PhD He is currently chairman of the Apple Education Foundation and a member of the initial executive committee of ASCILITE.

Computer Assisted Learning and the future of education

Brian W Carss
University of Queensland.

The changing nature of educational technology is towards a digital electronic environment, away from the present multiplicity of hardware. This change is expected to be completed within the next decade. The challenge that faces educational institutions during this period of change, is how to exploit the advantages that the new media offer and at the same time, enable academic staff to feel sufficiently comfortable with it, to want to use it. A "most likely" future is described and the implications of this to education are discussed.

This paper describes a research project which was carried out to determine what developments were likely in information technology and how would these developments effect education at the tertiary level. A Delphi survey was carried out with most of the major electronic equipment manufacturers being the principal respondents. The most striking finding of this survey was to confirm the rapid move away from film and analogue magnetic recording media to all electronic digital equipment. The research project was carried out in a number of specific phases. Phases 1 and 2 were devoted to a world-based Delphi study (Linstone, Turoff 1975) of media and educational technologists, designers and publishers with the specific aim of determining the state of educational technology by the year 2000 A.D.

Phase 3 was used to develop possible alternative future scenarios which incorporated the technological possibilities isolated during the first two phases. Some of these scenarios represent extremes in an educational continuum, and none of the proposed futures were seen as likely to evolve in a pure form. The most likely future will almost certainly be a weighted combination of the five scenarios.

During phase 4 of the project, two panels of senior educational professionals and practitioners engaged in a series of priority-setting sessions in an attempt to develop a set of weightings of influences on pre-tertiary and tertiary education in Australia for each of the scenarios.

The 5th and final phase was directed at developing a most likely composite scenario for education 2000 A.D. Details of procedures followed in the Delphi study, priority-setting sessions and the derivation of the composite scenario are described in Morris (1985). The results of the final phase only are presented.

Composite Scenario

Academic staff will still operate in 2000 A.D. in much the same way that they do now, but in a somewhat different style from that of the 1980's. Education will have become appreciably more sensitive to technological change by that time. The most significant change will be on the quality of education and the techniques used, brought about by the advances in technology.

By the year 2000 A.D., Society will be both dependent upon and dominated by, Information Technology (Goldsworthy.1980). Education being totally committed to the new technologies and their availability, will be equally totally dependent on and susceptible to the changes in technology itself.

There will be a strong growth in the acceptance of education as being a life long activity, with the consequent life-long demand for facilities, resources and opportunities. Because of the huge costs involved in the constant turnover of the latest technological advances, the institution will have to accept more responsibility for the provision of centrally produces programmes and resources of hardware and software. There will be a corresponding decrease in the individualistic approach to teaching.

Institutions

Institutions will have changed in that it will no longer be necessary for students to be on campus and attend formal lectures. Students may be required from time to time to attend specific lectures or participate in small group seminars but for the most part, students will rely on gaining access to the information resources they need through a highly sophisticated communications network which will enable them to work from their homes.

There will be increased governmental intervention into education in an attempt to link closely the products of the formal education system to societal demands foı increased sophistication in its workforce. So too, will there be a need to increase the quality of instruction. This will lead to the employment of teams of specialists in curriculum design and development, communication and evaluation.

There will be an identifiable increase in the influence on the curriculum of multinational and international corporations involved in the production of hardware and software. Such companies will exert pressure both overtly, through their products having been given an "international flavour", and covertly, through pressures exerted by the way of the "fashion" phenomenon operating through the cultural, entertainment and recreational needs of people in the workforce.

Students

Students are not seen as having a great deal of influence on educational decision-making. They will have a greater choice in the range of course offerings available to them but this will be matched by the increase in what is regarded as "core" material, dictated by a central authority intent on determining minimum standards of performance.

Lecturers

The changes for lecturers will be greater than those experienced by students. Most importantly, there will be an overall loss of prestige. Although the freedom to choose teaching strategies will be greater because of the greater range of techniques available, their influence on what is taught will be reduced. In other words, they will have less say in what is taught but more latitude on how to teach it.

A noticeable change that will have emerged will be the reduction in job security of academic staff. Increasing financial constraints will cause the Government as the funding agency, and the University as the employing agency to challenge the concept of tenure.

Instructional Designers and Other Specialists.

The increase in specialist influence on learning and teaching techniques will be profound. The control and dissemination of information will have become a highly specialized activity. Education will have adopted the "medical" model where each new development results in the emergence or retraining of specialists in the innovation. Educational specialization will range from Communications and information technology through curriculum design, monitoring and remedial specialities to those of the diagnostic and clinical educational physician.

There will be an increase in the production of one-way distributed media. It should be remembered that books, journals and all hardcopy materials fall into this category, as do films. As these new materials will be made available through the multipurpose media communication terminal, there is the opportunity for central control (censorship) to be applied.

A surprising growth area will be in the area of lecture room media. Despite a concern with the individual, large groups operating within lecture theatres will still be seen as appropriate for some kinds of learning. The cost of optical film-based materials will have increased to such an extent that digital electronic solutions will be sought using tape or disc. Long-life laser and digital technology, together with optical fibre transmission will see high quality audio visual distribution for display on large screens in lecture theatres. There will be an added dimension of interactivity brought to theatre presentations thereby upgrading the quality of instruction.

It is unlikely that the optical video disc will have a great impact on education because the cost of production and the specialized equipment needed for the manufacture of the discs. Thus education generally, will become increasingly a system which will use the latest information technologies supported by highly trained specialists in their application.

Implications of this Scenario for Teaching

To reiterate the implications of these technological developments, the complexity and sophistication of the new technology will require that it be handled by specialists. Specialists not only in the use of the hardware, but also specialists in the educational exploitation of the capabilities of the hardware.

Few tertiary institutions in Australia appear to have recognized the need to produce specialists in instructional design as being important for their inevitable futures. Concer for the development of CAL in a few institutions is in response to economic pressures which are present. It is hoped that by adopting CAL as a teaching medium, the institution can avoid increasing its staff to cope with the increasing numbers of students seeking tertiary education.

Several universities are experiencing a "blow out" in student enrolments in the Social Sciences particularly in Commerce and Economics. At the University of Queensland in the Faculty of Commerce and Economics the use of computer managed learning techniques (CML) is a large enrolment (700) introductory financial accounting subject has made it possible to more than halve the number of tutorial assistants needed as well as reduce the number of academic staff from three to one. This has been achieved without any reduction in the quality of instruction. It has however, placed very high demands on the academic staff in having to master the technology as well as be concerned about the restructuring of the course content. The use of CML demanded a restructuring of the subject as well as the creation of a sizeable database of test items. The items in the database had to be carefully linked to the specific objectives of the course. These were tasks that took many hours of academic staff time to complete.

The major impact of the use of CAL in this way has drawn attention to the need for staff development. Staff development in the use of CAL cannot be confined to learning of an authoring language or authoring system, but must also include developing a facility in instructional design. It is not unusual in the beginning for new CAL authors to believe that writing instructional material amounts to little more than the transfer of their lecturing notes onto the computer system. Many would be authors believe also, that it is sufficient to give students new information as written text on a screen followed by a question as the only teaching strategy. If the students get the correct answer, the lecturer assumes that the students know and understand the content. Such naivete has dogged the widespread adoption of CAL since the mid-1960's.

Clearly, would be authors need to be knowledgeable of teaching strategies such as Drill, Test, Inquiry, Simulation, Tutorial and Serendipity. These strategies can only be applied in an appropriate mix when the appropriateness of using CAL has already been determined. Unfortunately, the acquisition of teaching strategies is not a normal part of an academic staff members training, so unless academic staff make a deliberate effort to seek assistance, their teaching programs will be distinctly amateurish.

It would seem reasonable then for an institution which is committed to fostering better teaching, by whatever medium, should be concerned that its staff be properly trained in these techniques. Many universities in Australia already have organizational structures in place to carry out this kind of training. However, the effort will still be that of an individual lecturer.

In the commercial world, in contrast to the academic world, computer based training (CTB) is seen very much as a team effort. A team in which the content specialist may be a different person from the author. Such a team may consist of an author, a programmer, a content specialist, a graphics design artist and a psychologist. In business, where costs must be contained, a team approach to the development of computer based training materials is taken as a must.

If universities are going to take a lead from business, there is an obvious need to establish teams skilled in the use of CAL. A CAL Unit will have a more precisely defined set of responsibilities than the existing Higher Education or Tertiary Institutes. It will more often than not co-exist with a Tertiary Institute and make calls on its staff for their specialized knowledge in staff development and student assessment.

With the futures as described earlier, it is important that a highly skilled and knowledgeable team be assembled to assist university staff increase their effectiveness as teachers. For them, there is the added reward in having used CAL techniques in their teaching, of having more discretionary time to devote to individual students and their problems, as well as to research work.

References

Bell D., (Ed) (1968) *Towards the Year 2000: Work in Progress*, Boston: Beacon Press, pp 320.

Goldsworthy A. W., (1980) *Technological Change - The Impact of Information Technology*. Canberra: AGPS, pp 138.

Linstone H.A., Turoff M., (Eds) (1975) *The Delphi Method, Techniques and Applications*, Addison-Wesley, pp 620.

Morris M., (1985) *The Future of Pre-Tertiary Education in Australia.* Unpublished Ph.D. thesis, University of Queensland, pp 461.

Dr Brian Carss is a Reader in Education at the University of Queensland, where he is Dean of the Faculty as well as being the Coordinator of the CAL Unit. He has a particular interest in the application of computers to teaching. He has had a long standing association with CAL being one of the small group of staff at the University of Illinois who wrote instructional material for PLATO 1 during the early 1960's.

Demystifying tertiary education

Neil J. Clark
Faculty of Business Studies
Bendigo College of Advanced Education

Recent work in instructional psychology has strongly influenced the development of a new study strategies diagnostic and teaching program for tertiary students. The program will cover some hundreds of strategies, largely conceived of as academic tricks of the trade, and associated attitudes. In using the program students are asked if they use each strategy, or hold particular attitudes, via multiple choice questions. After responding they are provided with comments justifying preferred answers. Preliminary trials with both computer and booklet presented material have shown excellent acceptance by 112 C.A.E. students.

Tertiary education is a mystery to many students in the sense that they cannot cope and don't know why. For these students the following scenario is typical. They arrive at college or university after having negotiated twelve or more years of educational hurdles. Their expectations are frequently buoyant because they have demonstrated their competence, over and over, and there seems little reason to believe that success will not continue or that there is any real need to change their methods. Statistical evidence of high drop-out and failure rates (usually regarded as being somewhere between 40 and 60 percent), if known to students, is disregarded, the "it won't happen to me" belief prevailing. But then things start to go wrong. Reading becomes difficult, both in the amount required and in comprehending what is written. Lectures and tutorials are unfamiliar means of instruction. Written work which previously satisfied teachers is now criticised or rejected. Higher standards of spelling and grammar are applied. Work loads exceed those previously experienced. Responsibility for learning, previously borne by teachers, is now thrust abruptly onto students. As students struggle to cope, often with little guidance, they fall behind with work deadlines, with an associated decline in work quality. Failures and dropouts ensue. It is a picture familiar to any tertiary teacher as well as to student counsellors and, ultimately, to many students.

Secondary education is frequently blamed for not "preparing" students for tertiary studies, by the public as well as by academics. Even my corner deli. proprietor commented, "The schools don't learn them nothing nowadays". The students themselves are often accused of laziness while others blame poor standards of tertiary teaching.

There are clearly many places in the system where attempts at remediation might be undertaken. One of these areas, currently attracting interest, particularly in the U.S.A., (Weinstein & Underwood, 1985; Dansereau, 1985; Weinstein & Mayer, 1986; Paris & Wixson, 1983; Brown, Campione & Day, 1981; McKeachie, Pintrich & Yi-Guang Lin, 1985), is that of teaching students reading and study strategies. This paper gives some preliminary details of one such attempt in an Austral' n College of Advanced Education. Th. effort has attracted strong student support and staff interest but systematic validation studies have yet to be undertaken. The aim of the work has been to teach students strategies they presently do not use and to identify and at least partially remediate many of the defective methods and attitudes which prevent students from performing adequately. This is attempted via a questioning of students about their use of a large number of specific strategies, and of a lesser number of attitudes, and then, by commenting on their responses, to indicate the reasons why particular answers are preferred.

A computer presentation of the questions and comments has been favoured principally because of the control thus provided in evaluating the adequacy of the study strategies program and in making needed modifications.

Conceptual Framework

This work has been strongly influenced by the cognitive approach to learning where students are seen, not as passive recipients of information provided, but as active agents using previously acquired knowledge and skills to select and process parts of that information in line with their existing schemata (Anderson, 1985; Wilson & Anderson, 1986).

In the activities of selecting and processing students must use their working memories which have processing and storage functions competing for use of a limited capacity (Baddeley & Hitch, 1974; Case, 1978; Daneman, 1984). If capacity is taken up, for example, in attempting to decipher u familiar words in reading, there will be a diminished capability to keep track of the argument, to anticipate what comes next, or to ask oneself questions etc.

Students are also seen as having a capacity to reflect on their own processes, to be aware of the existence of various strategies and to monitor their success with strategies, changing them where necessary. Students are also viewed as having a capability to consider various options for organising selected input. Such metacognitive strategies (Brown, 1985; Brown, Armbruster & Baker, 1986; Lawson, 1984) have been shown to play a vital role in reading and studying. Study strategies are thought of as ways of undertaking specific tasks and very large numbers of such strategies, probably thousands of them, are thought to be involved in skilled performance as a student.

Skilled students, however, are not likely to be conscious of their use of well practised strategies since consciousness of skills recedes as they develop (Fitts & Posner, 1967). It is thus extremely difficult, in the case of intellectual skills, to find out how skilled performers actually operate. Further, even when a strategy used by a skilled person can be identified, it may not be possible to teach that skill directly to an unskilled person, because the latter may not possess necessary entering behaviours.

Skilled performance, at any given time, is not seen, however, as a simple function of individual learning (Vygotsky, 1978; Day, 1983). Performance takes place in a social environment from which varying amounts of assistance - from teachers, tutors, other students, counsellors, libraries and other resources - may be obtained. Reliance on these external resources for skill development diminishes as students internalise new skills ere is thus a need to ensure that external supports are appropriately provided and appropriately used - not too much, not too little - in the development and internalisation of skills.

But more than intellective strategies are involved in the success of students. There is also a need for hard work, for effective time management, and for appropriate attitudes, even attitudes about the nature of learning itself. There is also a need for an accurate grasp of the expectations of academic staff, for social skills in dealing both with fellow students and with academic staff, and for an understanding of the organisational setting within which students have to operate. Further, there is a need for students to learn strategies for self control and for handling stress and emotions.

No doubt many students succeed while exhibiting defects in some of these areas, perhaps by compensating with strengths in others. An example is the student who has poor abilities in many areas but who works very hard.

Students who have been involved with this program use the concept of "getting enough things right", and however imprecise, and ungrammatical, that expression may be, it probably captures the major requirement for success.

American study strategy programs, (Weinstein & Underwood, 1985; Weinstein & Mayer, 1986; Dansereau, 1985) have tended to concentrate on intellective type strategies such as reading comprehension and memory skills, whereas in this program a much broader view has been taken with cognative, affective and social skills being stressed in addition to cognitive ones.

Methodologically the program is seen as a multi-faceted, compound treatment, very much in the "shotgun" tradition. Such treatments are favoured by experienced teachers but largely ignored by psychological researchers because of the difficulty of sorting out the respective contributions of various elements in the treatment. See however, (Brown, Palincsar & Armbruster, 1984) who argue for a research strategy of first securing educationally relevant and transferable treatment effects by the use of multiple treatments and then, at a later stage, of attempting to isolate individual tr atment effects.

The items chosen for the program were expressed in question form, with students being asked if they used each strategy or, in the case of attitudes, held each attitude. Multiple choice options were provided for responses using a four category format. A questioning mode of presentation of material was chosen because of the known beneficial effects of questions in aiding reading comprehension, (Duffy, Roehler & Mason, 1984; Andre & Anderson, 1978) and because early trials using questions without comments had been well accepted by students.

It was thought that students using the program would identify some strategies which they were already using and gain added assurance and confidence thereby. Second, they would identify some strategies which would require little effort to adopt since most of the required entering behaviours were present. A third group of strategies encountered would require varying degrees of effort by students to learn and would be unlikely to be adopted without that effort. None-the-less, awareness by students that such latter strategies existed was thought likely to provide a series of quite specific goals for strategy learning by students. Quite apart from specific strategies, it was anticipated that working through some hundreds of strategies in a series of sessions would be likely to have a large and durable effect in that students would be far more likely to think about, question and monitor their methods than in the past. The need for students to think about their methods and about what they are trying to learn has been stressed by many writers, (Tobias, 1982; Day, 1980; Biggs & Rihn, 1984).

Selection of items for the program

Items were constructed using information gathered from the following sources:-

a. Strategies which have been used by researchers in reading and studying.
b. Strategies identified in previous study

strategies inventories (Brown & Holtzman, 1965; Dansereau et al, 1975).
c. Typical "How to Study" handbooks (Perry, 1983; N.Z. Council for Educational Research, Study Habits Evaluation and Instruction Kit, 1979; McEvedy & Jordan, 1986).
d. The literature on teaching reading strategies e.g. Herber 1978.
e. Discussions with and observations of students.
f. Discussions with academic colleagues about their strategies.
g. Introspection of my own strategies in reading and studying.

The search for strategies and attitudes was not restricted to efficient ones but also included several which have been identified as inefficient for students or appropriate only for an earlier stage of development. The "telling all" strategy for answering essay and examination questions, in which students simply write down all they know about topics, and the "copy delete" strategy for summarising texts, in which the text is simply copied less unimportant details, are examples.

Additionally, attention was given to strategies which have been studied with both primary and secondary students but which were thought likely still to be used by some tertiary students. The "telling all" and "copy delete" strategies are examples.

Finally, some strategies were included which were thought likely to be used by all students, to ensure that at least some of the methods about which they were questioned would be seen as appropriate.

The classification of strategies

The strategies and attitudes were classified into categories which were, for the most part, seen by students as major problem areas. The categories presently completed are:-

a. Learning new vocabulary - 60 items.
b. Preparing essays and assignments - 60 items.
c. Preparing for tests and exams - 60 items.
d. Taking examinations - 20 items.
e. Reading productively - 140 items.

Other categories planned include:-
f. Using tutorials to learn.
g. Learning from lectures.
h. Time management.
i. Improving motivation.
g. Managing stress and anxiety.

Preparation of comments

Comments were prepared for each question in the program explaining the reasons for preferred answers. The comments were written in a conversational manner and for the most part drew on empirical studies or psychological theory for their content. Where that was not possible comments were based on teaching experience. In some comments small amounts of information were included about empirical studies or basic psychological theory supporting the program. Important principles were repeated in comments on a number of related strategies.

Examples of questions and comments used in the program are shown in Appendix 1.

Presentation format

Three alternative formats have been planned for computer presentation to enhance the utility of the program both as a research and diagnostic instrument and as a teaching or tutorial device.

a. Questions only - for research purposes.
b. Questions only in blocks of 20, followed by a repetition of the questions, this time with each question being accompanied by a comment and the preferred answer. This format combines both re-

search and tutorial functions.

c. Questions, each with associated comments. This format would be used mainly for tutorial purposes.

Twenty question blocks, rather than some other number, were chosen to balance the need to provide reasonably short work periods against the need to obtain reliable data for analytical purposes and for reports to students.

In an early compute presentation of the program format (a) was used, while in later hard copy trials format (b) was used. In future computer presentations using formats (b) and (c), reports will be given to students after each block of 20 items. The reports will relate a student's score to established norms and recommend remedial actions. These might include reading further instructional material.(Clark, 1986a, 1986b) consulting a student counsellor or working through practice examples of new strategies etc.

Preliminary trials with the program

Almost all of the effort which has been put into this work so far has been directed at completing segments of the program and making these available to students many of whom are struggling to cope with their work. Initially a 417 item questionnaire was offered, both on computer terminals and on hard copy, to 80 second year Business Studies Degree students. Students completed the questionnaire outside of class time in 4-6 sessions as part of their course work in a one semester unit, Administrative Studies 1. Subsequently these students were required to write an essay saying what they saw as their major problems as students and what they proposed to do about those problems. Three hour long lectures and three hour long tutorials were used to introduce some of the psychological principles involved. Finally, a short questionnaire seeking opinions about what had been learned from the whole exercise was completed anonymously by these students.

All but two students regarded the exercise as a worthwhile one and indicated they wanted the work continued and extended. Specifically they wanted a one semester course in study methods introduced in the first year of their course. All of the students commented about their disappointment that most of the strategies covered in the program had not been taught in secondary schools and most claimed to have given little thought to their own methods hitherto.

Although a reduction from 50% (n=83) to 37% (n=88) occurred in the combined drop out and failure rate for this unit when compared with the previous year, results in other units suggested a better group of students was involved in the present year. Additionally other changes in the teaching a: examining of the unit were introduced wl n may also have had effects. It was thus not possible to assert, with any confidence, that the improvement cited above resulted from the use of the program.

The computer presentation used for this work was flawed by programming bugs, slow screen presentation times due to computer overload, poor segmentation of the program and, in a few cases, poor wording of questions and of multiple choice alternatives. None-the-less, despite all problems, the work was strongly supported by students who also provided encouragement for the development of the revised, categorised version of the program, including comments.

At the time of writing, July, 1987, 32 students have completed, on a voluntary basis, some or all of the first four categories of questions described earlier and other students are daily seeking access to the program. No proper analysis of the performance of these students has been made beyona that of checking that better students are performing better on the program.

The response of students to this second version of the program has been very enthusiastic, many of them claiming to have altered their study methods as a result of completing sections of the program.

Discussions with other staff members have suggested that there may have been some transfer of skills to other areas.

Future work

Besides working to complete additional segments of the program, work is now underway to set up the presently completed parts of the program on an I.B.M. System 2, Model 50 computer with Model 8514 high resolution colour screen, using the FORGE computer based learning software package. The latter provides for the design, layout, graphics, authoring, collation and management of an instructional program.

Once this work is completed, systematic trials will be undertaken to assess the worth of the program, to establish norms, to develop reports for students and to modify the program where required. Longitudinal studies to ascertain if study skills improve during the period of a degree course are also planned.

While it would be quite wrong for anyone to believe that a relatively brief exposure to this program would "solve" the many problems faced by students referred to earlier, it does seem likely that, with appropriate course integration, the program could find a place in, or as an adjunct to, study strategies teaching programs and student counselling in many tertiary and some secondary institutions.

References

Anderson, R.C. (1985) Role of the reader's schema in comprehension, learning and memory. In H. Singer & R.B. Ruddell (Eds.), *Theoretical models and processes of reading*, third edition, Newark, Delaware: International Reading Association.

Andre, M. & Anderson, T.H., (1978) The development and evaluation of a self-questioning study technique. *Reading Research Quarterly*, 4, 605-623.

Baddeley, A.D. & Hitch, G., (1974) Working memory. In G.H. Bower (Ed.), *The psychology of learning and motivation*. (Vol.8), New York: Academic Press.

Biggs, J.B. & Rihn, B.A., (1984) The effects of intervention on deep and surface approaches to learning. In J.R. Kirby (Ed.), *Cognitive strategies and educational performance*. Orlando: Academic Press.

Brown, A.L., Armbruster, B.B. & Baker, L., (1986) The role of metacognition in reading and studying. In J. Orasanu (Ed.), *Reading Comprehension: From research to practice*. Hillsdale, N.J.,: Lawrence Erlbaum Associates.

Brown, A.L., Palincsar, A.S. & Armbruster, B.B., (1984). Instructing comprehension fostering activities in interactive learning situations. In H. Mandl, N.L. Stein & T. Trabasso. (Eds.) *Learning and comprehension of text*. Hillsdale, N.J.: Lawrence Erlbaum Associates.

Brown, A.L. Campione, J.C. & Day, J.D., (1981) Learning to Learn: On training students to learn from texts. *Educational Researcher 10*, 12, 14-21.

Brown, W.F. & Holtzman, W.H., (1965) *Survey of study habits and attitudes*, (Form C). New York: The Psychological Corporation.

Case, R., (1978) Intellectual development from birth to adulthood: A neo-piagetian interpretation. In R. Siegler (Ed.), *Children's thinking: What develops?*

Hillsdale, N.J.: Lawrence Erlbaum Associates.

Clark, N.J. (1986a) Questionnaire on strategies for writing summaries. Unpublished paper, Bendigo College of Advanced Education.

Clark, N.J. (1986b) Strategies for locating and/or inventing topic sentences. Unpublished paper, Bendigo College of Advanced Education.

Daneman, M. (1984) Why some people are better readers than others: A process and storage account. In R.J. Sternberg (Ed.), *Advances in the psychology of human intelligence* (Vol.2). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Dansereau, D.F., (1985) Learning strategy research. In J.W. Segal, S.F. Chipman & R. Glaser (Eds.) Hillsdale, N.J., Lawrence Erlbaum Associates.

Dansereau, D.F., Long, G.L., McDonald, B.A. & Atkinson, T.R., (1975) *Learning strategy inventory development and assessment.* (AFHRL-TR-75-40, Contract F41609-74-C-0013). Brooks AFB, Texas, Air Force Systems Command.

Day, J.D., (1980) *Training summarisation skills: A comparison of teaching methods.* Unpublished doctoral dissertation, University of Illinois.

Day, J.D., (1983) The zone of proximal development. In M. Pressley & J.R. Levin (Eds.) *Cognitive strategy research.* New York: Springer-Verlag.

Duffy, G.G., Roehler, L.R. & Mason, J., (1984) (Eds.). *Comprehension instruction.* New York, Longman.

Fitts, P.M. & Posner, M.I., (1967) *Human performance.* Belmont, Calif.: Brooks Cole.

Herber, H.L., (1978) *Teaching reading in content areas.* Second edition, Englewood Cliffs, N.J.: Prentice Hall.

Herber, H.L., (1985) Developing thinking and reading skills in content areas. In J.W. Segal, S.F. Chipman & R. Glaser (Eds.), (Vol.1.). Hillsdale, N.J.; Lawrence Erlbaum Associates.

Lawson, M.J., (1984) Being executive about metacognition. In J.R. Kirby, (Ed.). *Cognitive strategies and educational performance.* Orlando: Academic Press Inc.

McEvedy, M.R. & Jordan, M., (1986) *Succeeding at university and college.* Melbourne, Thomas Nelson.

McKeachie, W.J., Pintrich, P.R. & Lin., (1985) Learning to learn. In G. d'Ydewalle. (ed.) *Cognition, information processing and motivation.* North Holland: Elsevier Science Publishers.

Paris, S.G., Lipson, M.Y. & Wixson, K.K., (1983) Becoming a strategic reader. *Contemporary educational psychology, 8,* 293-316.

Percy, D., (1983) *Study tactics.* Melbourne, MacMillan.

Raphael, T.E. & Gavelek, J.R., (1984) Question-related activities and their relationship to reading comprehension: some instructional implications. In G.G. Duffy, L.R. Rochler & J. Mason (Eds.) *Comprehension instruction.* New York, Longman.

Study habits evaluation and instruction kit. (1979) Wellington, N.Z.: New Zealand Council for Educational Research.

Tobias, S., (1982) When do instructional methods make a difference? *Educational Researcher, 11,* 4, 4-9.

Vygotsky, L.S., (1978) (Mind in society: The development of higher psychological processes) M. Cole, V.J. Scribner & E. Souberman, (Eds. and trans.) Cambridge, Mass: Harvard University Press.

Weinstein, C.E. & Mayer, R.F., (1986) The teaching of learning strategies. In M.C. Wittrock (Ed.) *Handbook of research on teaching,* third edition. New York: MacMillan.

Weinstein, C.E. & Underwood, V.L., (1985) Learning strategies: In J.W. Segal, J.W., S.F. Chipman, . & R. Glaser, (Eds), (Vol.1.). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Wilson, P.T. & Anderson, R.C., (1986) *What they don't know will hurt them: the role of prior knowledge in comprehension: From research to practice.* Hillsdale, N.J.: Lawrence Erlbaum Associates.

Appendix 1: Examples of questions and comments used in the study strategies program

Q.1 I believe my teachers and lecturers, rather than me, are responsible for teaching me any new words I need to know:

(a) Yes, I strongly believe that.
(b) Yes, I believe that.
(c) No, I don't believe that.
(d) No, I strongly oppose that.

COMMENT: Most lecturers and teachers would be likely to explain important new technical concepts. But they are unlikely to have enough time to deal with every new technical term. Additionally, they are unlikely to spend much time in teaching you many non technical words. So, for the most part, learning new words is up to you.

The preferred sequence of answers is therefore d,c,b,a.

Q.2 I set myself a target of learning a certain number of words every day:

(a) Yes, that is true for me.
(b) Yes, I do that every so often.
(c) No, I rarely do that.
(d) No, I have never done that.

COMMENT: The importance of a large vocabulary and its connection with your ability to read and write well, has been stressed in several other comments. So a definite plan to increase your vocabulary is an excellent strategy, provided that correct strategies are used (see other strategies for hints) to ensure that such learning is effective. Think of this: 10 new words a day would mean 3650 new words a year, and over 10,000 new words over the three years of a degree. Such an increase in vocabulary could be expected to make your work as a student a lot easier and to improve your performance.

The preferred sequence of answers is therefore, a,b,c,d.

Q.3 When I am reading for an essay, assignment or report I ask myself frequently, (at the end of paragraphs, sections, chapters etc.) "What is the author's argument in a nutshell in this part?" I do this:

(a) Almost always.
(b) Most of the time.
(c) Sometimes.
(d) Almost never.

COMMENT: This is a very useful summarisation strategy and a good way of fitting in new information with your existing knowledge. The action of summarising, in your own words, which is integral to this strategy, helps tie in the new with the old. The strategy is also a precursor to the strategy of keeping a running check in your mind of a developing argument, where, with each new paragraph, you run over in your mind the sequence of argument up to that point.

The preferred sequence of answers is therefore a,b,c,d.

Q.4 If I do poorly in an essay, assignment or report I consult my lecturer about which students did well and ask him to arrange for me to read two or three of the top efforts so that I can note differences in standards and in approach to aid me in future efforts. I use this strategy:

(a) Very frequently.
(b) Frequently.
(c) Sometimes.
(d) Almost never.

COMMENT: Students often learn things better from each other than from their lecturers. Perhaps it is because there is a more relaxed relationship between students. Whatever the reason it makes sense to try to get a line on differences in standards so that you know just how much, and in what directions, you need to improve.

The preferred sequence of answers is therefore a,b,c,d.

Q.5 I think the sole purpose of essays, assignments and reports is to test the knowledge of students of course content in particular subjects:

(a) Yes, I am quite sure that is true.
(b) Yes, I am fairly sure that is true.
(c) No, I don't think that is true.
(d) No, I am quite sure that is not true.

COMMENT: It certainly is true that essays etc. test your knowledge of different subjects. But they often have another objective which is sometimes not fully grasped by students. That objective is to see if students can handle the analysis and discussion of complex verbal issues with the required skill and whether they demonstrate the required depth of thought seen as appropriate at a tertiary level. Now, if you don't know of the existence of this second goal there is a fair chance you won't give it the required attention - with dire consequences.

The preferred sequence of answers is therefore d,c,b,a.

Q.6 When I have to remember different theories or approaches to a topic for a test or examination, I draw up tables listing important criteria or characteristics so that they can be readily summarised and compared. I do this:

(a) Very frequently.
(b) Frequently.
(c) Sometimes.
(d) Almost never.

COMMENT: This strategy is one of the most useful of all for tertiary level students to learn. The strategy allows good summarisation, comparisons and helps memory. An extension of the strategy allows subcategories in one or both sides of the table.

The preferred sequence of answers is therefore a,b,c,d.

Neil Clark is a psychologist and senior lecturer in organisational behaviour and administration at the Bendigo C.A.E. He believes that C.A.L. will be particularly useful in the development and testing of his tertiary level study strategies program and of a similar program for use with secondary students.

Computer enhancing using a function graph plotter

Debbie Clayton, Department of Mathematics and Computing
Phillip Farrands, Mathematics Learning Centre
and Matthew Kennedy, Department of External and Continuing Education
Capricornia Institute

Computers are a valued resource in the instructional process, however they need to be used intelligently. One way of doing this is to use the computer as an experimental tool, to centre the learning activity on the student and provide a means to aid students' understanding of concepts. This paper details how a team from Capricornia Institute developed Cap-Graph, a sophisticated graph plotting package. This package is an experimental tool which, with appropriate worksheets, can be used by a wide range of students to develop and deepen conceptual understanding in their mathematics courses. Cap-Graph has been trialed in a laboratory situation with an internal mathematics subject for biology students. The results of this trial and the implications for incorporation of Cap-Graph experiments in other mathematics courses is discussed.

At last year's CALITE conference, one of the authors, Debbie Clayton, spoke about CAL trends in tertiary mathematics that she had observed in the UK and USA. One trend in undergraduate mathematics was away from the traditional dialogue/tutorial mode of the 1960's and 1970's to Computer Enhanced Learning (CEL). This term was coined by Professor Avi Bajpai of the University of Technology in Loughborough (Bajpai, 1985).

In a CEL application, the computer is put to "best" use by being either:

- an electronic blackboard,
- a numerical utility, or
- an investigational/experimental tool.

When the machine is used for investigation/experimentation, sophisticated pieces of software are used as tools in a laboratory setting to assist the student to understand mathematics. Worksheets or experiment sheets are used in the laboratory to guide the student interaction with the software.

This paper describes the development and use of such a software tool. Cap-Graph is a function plotting package, which, with appropriate worksheets has been trialed at Capricornia Institute to assist students studying an introductory mathematics subject for Biologists. The potential for the use of a piece of software such as this in other mathematics subjects is also addressed.

Rationale

Why develop a graph plotting package? An easy to use function plotting program, can be employed by students studying a wide variety of subject areas. For example, a Calculus student can use it to plot a function with a discontinuity when studying limits. A Statistics student can plot probability density functions with different parameters on each graph. This wide range of usage options, as well as the plotter's capability to graph quite complex mathematical functions makes it an "intelligent" choice of software tool.

The provision of such a tool to students was seen as one approach to assisting them in more easily attaining a holistic understanding of the graphical behaviour of functions.

Cap-Graph aims to assist students to learn their mathematics by presenting visual representations of the functions that they are studying.

The Software

Design Considerations

The Cap-Graph software was developed utilising the authors different areas of expertise. Defining the goal of the project and general system specifications was undertaken by the academic members of the team, whilst the programmer wrote the software. Close consultation was maintained between all three members of the team.

The program was written in Turbo Pascal, and extensive use was made of the Turbo Graphics Toolkit. Turbo Pascal was chosen to allow easy portability to a variety of machine configurations. The use of the Graphics Toolkit reduced development time in creating the user interface.

One of the aims of the system design was to produce a program which could, or be easily converted to, run on both NEC III and IBM compatible computers. The Capricornia Institute has a laboratory of 20 NEC APC III computers. In addition, similiar machines are installed in the Capricornia Institute's External Study Centres (13) located throughout Queensland. Many students own or have access to machines with IBM compatible graphics. Although NEC APC III computers are MS-DOS machines, the graphics are different to the de facto IBM standard graphics. Versions of Cap-Graph have been developed to run on machines which are compatible with the following:

- NEC APC III
- IBM PC, XT, AT with one of the following graphics cards
  CGA (Colour Graphics Adapter)
  EGA (Enhanced Graphics Adapter)
  Hercules (Hercules Graphics Board)

User Interface

It was necessary to ensure that the software was simple to use, to try to minimise student time required to become familiar with the program. The target audience for the program may not have touched a computer before, and students may be using the program away from any immediate assistance.

To achieve the aim of creating a software package which was easy to use and hence aid students learning; screen layout, user control, feedback and on-line help were all considered an essential part of the user interface. Possible outcomes of poorly designed or inappropriate interfaces are stated by Price (1985) as :

- the user refuses to make use of the system after initial testing;
- the user learns a few commands 'parrot fashion' and relies on using this, hence he/she cannot make use of all the system's facilities; or
- an intermediary is used who then becomes the local system expert.

Screen Design

Research into colour screens indicates there should be a strong contrast between foreground and background colour intensities (Issacs, 1986). Research has also been reported on colour combinations which meet a minimum criteria for legibility (Oliver, Sullivan & Barr, 1982). The colour combinations used in Cap-Graph comply with these findings.

The screen has two main functional areas, which are used consistently throughout the program (Godfrey & Sterling, 1982). The top line of the screen is displayed as a blue foreground on a yellow background. All command key options and error messages are shown on this line. The remainder of the screen has a yellow foreground on a blue background. These colours also provide good contrasts on monochrome displays. Main menu options are shown on the second line.

The screen design is basically one of a split screen with pop-up windows. Commands are listed at the top of the screen, and the graph displayed on the remainder of the screen. Pop-up windows are used to display sub-menus, help screens, user inputs and feedback on invalid syntax of user entered functions. Sub-menus, displayed in pop-up windows as required, occur at the top of the screen. Help is available throughout the program and is displayed in a pop-up window at the bottom of the screen. Current research tends to favour the use of pop-up windows to keep the screen uncluttered (Issacs, 1986).

User Control

Control of the program is maintained by the user with the cursor keys, the Return and Escape keys, and the Help or F1 key. Escape is always used to return control to the user by aborting the current process and returning to the main menu. The Return key is always used to accept input or select the highlighted menu option. The position of these keys (and different return key symbols) on the keyboard are explained at the beginning of the program. Menu options can be selected using either the cursor keys or the unique letter indicator for each option.

Feedback

Feedback to the user is provided during the graph drawing/calculating process by a small line which moves across the top of the screen. Explanatory feedback is given when invalid syntax in mathematical expressions is used.

Help

A detailed help facility has been designed as an integral part of Cap-Graph. Help provided depends on the users location in the program. Context sensitive help, available at all times, is considered essential in CAL programs (Hartley, 1986). Approximately 60 different help screens have been included in the program.

Program Features

Equations are entered in functional notation form. The scaling of the graphs is defined in terms of domain and range, and scientific notation is used, by the program, for large numbers.

Functions are entered by initially defining the dependent and independent variables. These are restricted to the English alphabet. In addition, the irrational number e is defined as a constant, so use of this letter is not encouraged. The program can store 26 different functions. The screen, however, is never automatically cleared so an infinite number of equations can be displayed at one time by editing then drawing the functions. Any of the previously defined functions may be edited by attempting to enter a function with the same dependent variable. A facility also exists to change the most recently entered, or edited function.

A wide variety of mathematical functions can be drawn using Cap-Graph. Both continuous and discrete functions can be displayed. Discrete functions are entered using the predefined integer component function. It is possible to display Binomial, Poisson, Normal and Exponential distributions using this approach. Taylor's Series approximations can be easily shown and the improvement in the approximation with each successive term clearly demonstrated.

Cap-Graph can also draw either the first or second derivative of the most recently entered, or edited function.

Defined Operators, Functions and Constants

The normal mathematical operators are available, namely:

+	addition
-	subtraction
*	multiplication (unnecessary if unambiguously implied)
/	division
^	exponents (5 cubed is entered as 5^3)
!	factorial (valid for whole numbers).

Functions included are:

sin(x)	trigonometric sine
cos(x)	trigonometric cosine
tan(x)	trigonometric tangent
arcsin(x)	inverse trigonometric sine
arccos(x)	inverse trigonometric cosine
arctan(x)	inverse trigonometric tangent
sinh(x)	hyperbolic sine
cosh(x)	hyperbolic cosine
tanh(x)	hyperbolic tangent
arcsinh(x)	inverse hyperbolic sine
arccosh(x)	inverse hyperbolic cosine
arctanh(x)	inverse hyperbolic tangent
exp(x)	exponential function (e^x is also valid)
ln(x)	natural logarithm
log(x)	common logarithm
sqrt(x)	square root
sqr(x)	squared function
rad(x)	converts the operand from degrees to radians
u(x)	heaviside function
int(x)	integer component
fract(x)	decimal component
round(x)	rounds to the nearest integer
abs(x)	absolute value

Constants

The constants e = 2.718281828
pi = 3.141592654 have been defined.

One of the more powerful features of the program is the ability to define a function in terms of previously defined functions. That is, if the functions f(x) and g(x) have been entered, the the following functions are examples of valid functions

h(t) = f(t) + g(t)   m(s) = f(g(s))
p(x) = abs(f(x)+g(x))

Scaling

The scale of the graph is user defined as a minimum and maximum for the domain and range. In addition, there is an autorange facility which will calculate a range so that at least 90% of all functions stored in the system will be displayed on the screen. Scaling is limited to numbers between +/- 999999999 to +/- 0.000000001.

Zoom Facility

A facility has been built into the program to allow close examination of parts of the graph. This option allows the user to zoom in on a region of the screen. The magnified section is displayed in a window on the screen which can be moved so that the original graph is not obscured.

The coordinates of the centre of the window are displayed, (to ten figures) at the top of the screen together with the width of the window, for example +/- 0.33333. Using the coordinates it is possible to find points on a graph to a maximum 10 significant figures.

Magnification of an adjacent region can be easily displayed using the cursor keys. The effect of this facility is similiar to moving a magnifying glass over a piece of paper.

Magnification of the zoom window can be reset at any time. The centre of the zoom window is set by defining the coordinates or by using the cursor keys.

Options

Hard-copy printouts of the screen can be obtained from Cap-Graph. Six different Epson printer modes are supported which will print the screen in different sizes and height-width proportions.

Computer Enhancement

The development of the software is perhaps the easiest part of computer enhancement. The enhancement of a subject (once the software tool exists) proceeds in the following manner:

- The syllabus is reviewed. Areas where students experience difficulty with their mathematics and could be assisted by a visual function representation are identified.
- In these areas, student centred interactions with the software are constructed by writing guided experiment / worksheets that can be used in a laboratory setting.
- A possible worksheet structure is:

  the aim/s of the experiment

  the necessary mathematical background the student needs to proceed with the experiment

  reference/s to appropriate texts

  instructions related to what activities the student should go through with the software

  student questions related to those activities.

Designing these experiment/worksheets so that the desired learning outcomes result for this guided interaction is rather difficult. For this, educators need to develop special skills related to the usage of this new teaching technology.

The Trial

The plotter and as[illegible]ed worksheets have been trialed by an internal first year biology degree mathematics class. During the semester these students worked through four experiment sheets in a labora tory setting. Instructors (the authors) were present to assist with both mathematical questions and questions about the software. The four worksheets for the experiments related to the topics:

Linear and Quadratic Functions Limits and Continuity Exponential and Logarithmic Functions Derivatives and Curve Sketching.

As an example of the type of activities in these interactions, consider the Quadratic Function section of the first worksheet. The aim of this experiment was to develop the student's appreciation of the effect that each parameter in a quadratic function has on the shape of its graph. Here students were first asked to plot a parent graph of x squared. Then, in a systematic way, the parameters were changed and the student was required to write down statements on their worksheets as to the effect that each parameter had on the shape of the graph.

At the end of the semester, the computer enhancement using Cap-Graph and the guided experiment sheets was evaluated by questionnaire. Eleven of the class of nineteen students responded.

It was found that, in general,

- the students liked going to the microcomputer laboratory and working through the experiments with Cap-Graph. They found the program easy to use, and felt positive about obtaining a graphical representation of the functions they were studying.

- those students who had not used a computer prior to the interactions, now felt more confident about using the computers, and
- the students considered it did help them to learn their mathematics.

The students also made suggestions as to how the user interface of Cap-Graph, and how the format/content of the worksheets could be improved. These suggestions were incorporated where feasible.

A Wider Audience

After the initial development and program testing with an internal class, Cap-Graph was made available to a wider audience. At the beginning of the second semester, 1987, copies of the software were located in external study centres and in the Library at the Capricornia Institute. A number of staff access the program using personal computers on their desks. Whilst direct incorporation of Cap-Graph as an electronic blackboard (in a lecture environment) has, as yet, not occurred, overhead transparencies of printouts from the program have been used.

The authors are currently writing a series of experiment sheets to be combined into a booklet form. This material will be available to students as resource material with the software in the Library and in external study centres. Lecturers, for a variety of subjects, will also be able to direct students to relevant worksheets as either an adjunct or as a formal component of the subject.

Portability of the software has meant that compromises needed to be made in the program design and features included. The main constraint is restricting the memory usage to 256kb. Since many users will have access to compatible computers without this restriction, a more sophisticated version is currently being planned. Features being considered for this package are:

- different colours for different graphs. This will not be possible on machines with CGA or Hercules graphics.
- a system of being able to match the graph to the function after the function has been drawn,
- an option to allow the definition of constants,
- displaying the equation in the normal mathematical form after it has been entered, exponent above the base, numerator over the denominator.
- a choice of scale to allow log-log and semi-log graphs to be drawn,
- incorporation of a numerical integration routine.

Feedback from academics in disciplines other than mathematics indicates that the scope for use of a function plotter is widespread. Cap-Graph is capable of being used as a tool in diverse areas such as engineering, financial mathematics, economics, and the behavioural sciences.

Summary

This paper has outlined the development and trialed use of a software tool that can enhance student learning of mathematics.

Software tools such as Cap-Graph do have considerable potential in not only mathematics teaching, but also in several other disciplines. The scope of its use is wide, limited only by the imagination and ability of the instructor to construct experiments/ worksheets to guide student interactions to achieve desired learning outcomes.

The authors hold that constructing software tools which can be used by a variety of students and in a variety of subject areas is an intelligent use of CAL in the 80's.

References

Bajpai,A. et al (1985), Mathematics and the Micro. Some hints for software development. *International Journal of Mathematics Education in Science and Technology,16*(3),407-415

Godfrey,D. & Sterling,S. (1982). *The Elements of CAL.* The How-to Book on Computer Aided Learning. Reston Va, Reston Pub. Co.

Hartley,J.R. Placing Expertise in Computer Assisted Instruction. *Fourth Calite Conference Proceedings,* Adelaide 1986,1-10

Issacs,G. Screen design fcr Computer Assisted Learning. *Fourth Calite Conference Proceedings,* Adelaide 1936,147-157

Oliver,D.,Sullivan,C.,& Barr,J. (1982). *So you want to be a Prestel Information Provider? A manual of practical advice for providing local community information on Prestel.* Aslib London, British Library Research & Development Report No. 5655

Price,J.A. Software Engineering : A Guide for CAI. *Third Calite Conference Proceedings,* Melbourne 1985,44-60.

Debbie Clayton, Lecturer, Computer Enhancement of Tertiary Level Mathematics, Remedial Mathematics, CAL User Interface Development.

Phillip Farrands, Lecturer, Screen Design, Secondary/Tertiary interface mathematics, effective use of computers in education.

Matthew Kennedy, Programmer, Creating effective software tools for student use.

Updating Mathematical skills: An evaluation of a computer based mathematics course

Michael J. Crock, Division of External and Continuing Education
Darling Downs Institute of Advanced Education

Updating Mathematical Skills is a continuing education course for students who wish to study at the tertiary level but who do not feel confident with their level of mathematical ability. The study materials consist of a study book, the set text "Self-Paced Introductory Mathemati~ , and an introductory audio cassette. All assignments for the course are completed on student answer sheets and computer marked with full feedback printed at the time of marking. Results and feedback are then forwarded to the student. Student progress records for all assignments are computer managed. The course has been fully operational for over 18 months with enrolments to date being over 300 students. Initial evaluation of the course shows a cost-effective utilization of computer managed learning, and a positive response to the course structure and effectiveness by students. An interactive version of assignments is currently under trial.

In any teaching-learning situation a large proportion of the teacher's time is devoted to prescribing learning activities, marking assignments, providing associated feedback and keeping records. This feedback usually includes advise on learning problems that have been diagnosed in light of student performance. A significant part of these activities can be assigned to the computer thus providing the basis for a computer managed learning system. In 1983 a computer managed learning system was established at the Darling Downs Institute of Advanced Education and the essence of this computer managed learning is represented in Figure 1. (Barker, White and Taylor, 1984).

Incorporated in the CML system is the development, by unit teams, of computer based LEARNING activities which provide students with opportunities to test the effectiveness of their learning and obtain grades and feedback. The 'managed' aspect of the system maintains records of students' use of CML activities, scores student learning activities and provides this data to lecturers.

The breadth of the CML activities provided in the system range from 'conventional' CML, with its emphasis on management of student records and scoring, to activities more representative of computer assisted instruction (CAI), involving extended feedback, branching programs, algorithmization, and the linking of microcomputers and videodisc systems.

To effectively utilize this CML system, the unit team needs to analyse and structure the subject matter in question into appropriate instructional modules, which require:

1. the precise sequencing of instructional objectives, which clearly define expected student learning outcomes;

2. the careful selection of associated learning activities, which clearly define student work programs; and

3. the construction of tests, which aim to measure the extent to which students have achieved each of the stated objectives. (Barker, White and Taylor, 1984)

By making this initial commitment to structure the subject matter in this manner, the unit team can then expect to provide the students with the following potential benefits:

1. early identification and resolution of problems — the pinpointing of student difficulties and the availability of this information to both students and examiners alike should lead to a rapid resolution of problems;

2. a clearly defined pacing mechanism — by meeting the deadlines set for the completion of each test, students can be sure that they are working through the materials at an appropriate pace;

3. immediate individualised feedback — an assessment of the extent to which stated objectives have been achieved; diagnosis of difficulties and associated provision of a list of learning activities which should lead to improved performance. (Barker, White and Taylor, 1984)

The Updating Mathematical Skills course which is to be evaluated in this paper was developed using the Darling Downs Institute of Advanced Education's CML system and unit team approach as outlined above.

Updating Mathematical skills

Updating Mathematical Skills is intended for students who wish to study at the tertiary level but who do not feel confident with their level of mathematical ability. It can be taken individually as a continuing education course, or it can be taken as a component of the Preparatory Studies Programme, which is a currently operating external programme developed to provide an alternative means of entry to tertiary courses for mature age students. Using the concepts of self-paced instruction, and mastery learning the course guides students through a carefully sequenced series

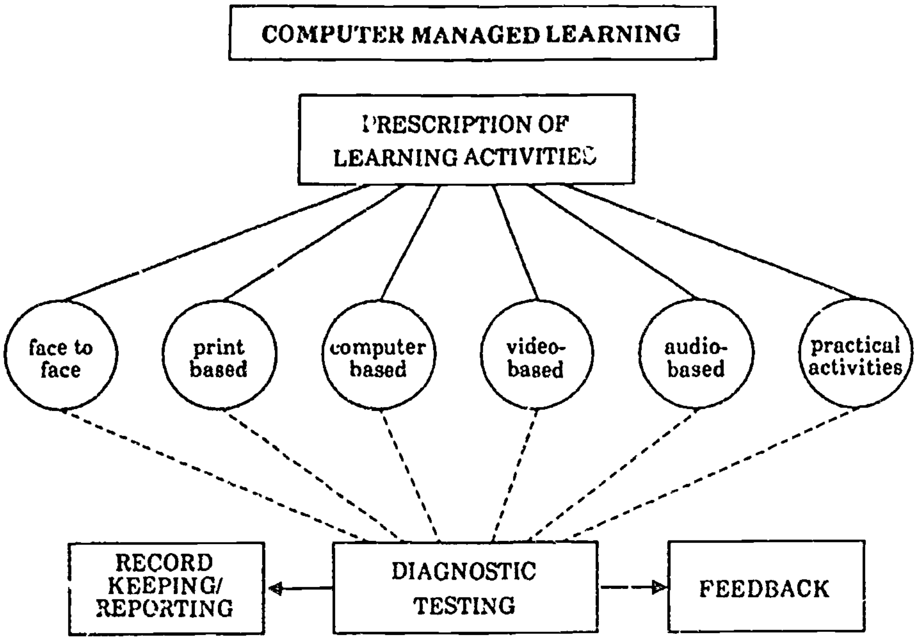


Figure 1: Key elements of Computer Managed LEarning

of topics which provides the foundation for understanding the mathematics that will be encountered in tertiary studies.

The self-paced structure of the course allows students to work through the material at a pace suitable to their needs, permitting them to work quickly through familiar materials, as well as allowing them the opportunity to seek additional assistance in areas of uncertainty. The mastery approach ensures that students successfully achieve the objectives of each topic before progressing to the next topic, which builds upon on the earlier material.

The main objectives of this course are:

1. To increase the student's understanding and confidence in mathematical skills to a level which will enable a student to successfully deal with the maths they will encounter in their tertiary studies; and

2. To develop a positive attitude towards mathematics which will assist the student in their understanding of basic mathematical concepts.

The course material includes:

1. An introductory audio cassette;

2. The set text: SELF-PACED INTRODUCTORY MATHEMATICS (SPIM) by Dobson and Stokoe;

3. Updating Mathematical Skills Study Book;

4. A set of assignment labels for CML activities.

The introductory tape gives students a brief introduction to the course as a whole, highlighting the important points they should be considering as they work through the course, and introduces students to a few of the team of course tutors who will be monitoring and assisting their progress.

The set text (SPIM) contains the required content for the course and the study book is provided to guide students through the material by introducing topics, highlighting problem areas, and providing additional worked examples and practice problems.

The structure of the course is based on the structure and sequence provided in the set text SPIM. Originally produced to provide the basis for a mathematics course for environmental science students, the style, content and level of the material presented in SPIM has been used to provide a solid mathematics base for students enrolled in a wide variety of courses including science, business, and engineering tertiary courses.

The study book, which has been specifically designed to enhance the material and facilitate the external teaching of the course, guides students through SPIM's 14 units, which have been grouped into six modules according to content. With the exception of the single unit Module 1, each module contains a series of two or three related units. The assignments for this course correspond directly to the modules.

The following figure illustrates the breakdown of the 14 units into the six modules, and briefly lists the content of each unit.

MODULE	UNITS	CONTENT OUTLINE
1	1	Number systems and elementary arithmetic
2	2	Notation, order and inequalities
	3	Graphical representation and equations of lines
	4	Sets
3	5	Polynominals and use of the binomial theorem

	6	Functions
	7	Inverse f .ctions and graphing of functions
4	8	Exponential and logarithmic functions
	9	Trigonometric functions
	10	Limits of sequences and functions
5	11	Introduction to differentiation
	12	Applications of differentiation
6	13	Introduction to integration
	14	Techniques and applications of integration

Figure 2: Course Structure and Content

The sequence of the units has been carefully planned to ensure that students progress through the material in a logical building block fashion.

Assessment

The assessment in the course is of two types:

1. Self-Assessment which takes the form of pre-tests, exercises, and sample quizzes in the study book and the set text SPIM. Such self-assessment items in this course are not required to be submitted and are for the students' benefit to show them how well they are understanding the material. The answers to all self-assessment items are given to students in either the text or the study book.

2. Formal Assessment, which takes the form of six CML module assignments. Each of these assignments has distinct sections for each unit of the text. (i.e. the assignment for module 2 has three distinct sections, one for each of units 2, 3 and 4 of SPIM.)

Appendix A contains an example of a CML assignment and the marking sheet the student would be expected to complete.

In order to successfully pass this course, students need to complete all six assignments and achieve a pass mark for each one. Assignments take the form of computer-marked tests and are only graded pass or incomplete. Student are always given feedback on incorrect answers and should they have an assignment marked incomplete, they are required to submit one of the two remaining assignments for that particular module, thus enabling the student to upgrade their standard of understanding to a pass level.

Appendix B contains an example of a student's marked assignment and feedback specific to the responses the student has answered.

The difficulty, style and length of assignments corresponds closely with the scope and content of questions found in the pretest and sample quizzes in the study book and text.

The CML system permits answers to be multiple choice or exact answer, and the feedback given can be specific to an exact answer or to a particular numerical range.

If students receive a pass for a particular module assignment, but for their own personal benefit would like to submit any of the remaining parallel assignments, this may be done and the results do not affect the pass mark already achieved.

Appendix C contains an example of the student record report used to monitor student achievement levels and progress through the course.

Student enrolments

Although the teaching year at Darling Downs is divided into three periods, students may enroll in Updating Mathematical Skills at any time, as either a Continuing Education course or as part of the Preparatory Studies programme, allowing maximum flexibility. However, the final examination for the Preparatory Studies students is held only three times per year, in June, November and February. Students study at their own set pace, however, a student who wishes to study at about the same rate which is expected of people enrolled part time in accredited courses, such as the Bachelor of Business or the Associate Diploma in Engineering, should complete the course in approximately 18 weeks.

During 1986, 194 students enrolled in the Updating Mathematical Skills course as Continuing Education students. Of this number 17 have successfully completed the course, while 152 are currently still actively enrolled and continuing studies. In this same year, 238 students enrolled in the course as Preparatory Studies students. Of these, 20 have successfully completed the course and gained entry to accredited courses, and 92 are still actively enrolled.

So far in 1987, there are 106 students enrolled as Continuing Education students and 207 students taking the mathematics course as part of Preparatory Studies. Out of this enrolment, 25 have already successfully completed the course, with 24 of these gaining entry to accredited courses.

The vast majority of both the 1986 and 1987 students are mature age students who are interested in returning to tertiary studies after a significant number of years since last attempting studies of any description. Illustrating this point is an enrolment survey which shows that average time since last studying mathematics for the current group of over 300 students is 14.6 years, with individuals ranging between three months and 46 years since last studying in this area. The majority are also interested in attempting external studies while maintaining their current job position. While most students live in Queensland, there are also a number of interstate and international students registered in the programme. (Crock, 1987)

To support and run the programme with these numbers there are three part time tutors in the Updating Mathematical Skills course. The course also utilizes Continuing Education administrative staff and DDIAE administrative services such as mailing, student enrolments, student support centre, and advertising.

Methods of evaluation

Students are given the opportunity to complete written questionnaires evaluating each component of the Updating Mathematical Skills course, and in fact comment on each assignment and its related material. Appendix D contains a sample of the evaluation form used to collect data. The appropriate course development unit team then has the task of recommending programme improvement based upon students perceptions plus the team's evaluation of the students standards of achievement. In essence, this means the current evaluation of the course with reference to meeting the previously stated objectives, is based upon student's direct evaluative responses and student's achievement records. Subsequent improvements to the material are then incorporated in to the material where considered appropriate by the leader of the course development team. This process is carried out on an annual basis.

Results of evaluation

Dealing first with student evaluation of the course, Table 1, on the following two pages, presents the results for the opinionnaires for modules 1 to 6. Responses range from "strongly agree" (=1) to "strongly disagree" (=4) and the table shows the mean response (M), the sample size (N) and the standard deviation of responses (SD). The final question listed on the table deals with the average number of hours required to work through a particular module.

Table 1: Results for Student Evaluation.

Evaluation based on student success rates is limited at this stage as the course is still relatively new and there are still a large number of both 1986 and 1987 students still actively completing the course. However if you examine the subsequent success rates of students who have continued on to accredited courses, having first successfully completed the Updating Mathematical Skills course, it appears the course has provided a solid mathematical background, and signalled that students would be able to cope with tertiary level studies. Of the first 16 accredited course units(many containing mathematical components) attempted by successful Updating Mathematical Skills students, the results obtained were: five As, six Bs, four Cs and only one incomplete.

Discussion

Evaluation forms submitted by students have been strongly in support of the Updating Mathematical Skills course with respect to its format and perceived applicability in achieving the set objectives of the course. In particular, while it was expected students would find the course to become slightly more difficult as they progressed through to the latter stages, (reflected in responses to Question 2 of the evaluation) the students maintained a high level of interest throughout the course, shown in the responses to Question 1 of the evaluation.

Results for all sections of Question 3 illustrate the students' support for the self-paced instructional package they were required to use, particularly in the first three modules evaluated. The results for the last three modules still reflect students' strong support, yet reflect the increasing difficulty they must have experienced as they progressed through to the more difficult concepts.

Questions 4 and 5, dealing with the appropriateness of the assessment items and the services provided by the DDIAS, illustrate a high level of support for the computer based assignments, and the associated marking procedures including the extended feedback provided.

The final question asked of students related to the number of hours spend on a particular module. If you take into account the fact that Modules 2, 3 and 4 encompass three units in the set text, and 5 and 6 two units, the average number of hours per unit in the text range from 6.55 hours for units in Module 2 to 14.18 hours for units in Module 6. These are encouraging results as it is stipulated that the average time required to be spent per unit should, on average, be in the vicinity of 8-10 hours.

As stated earlier, discussion on the effectiveness of the course based on pass rates is at present limited. While tentative conclusions that the course is achieving satisfactory results can be made, statistics need to be finalized after the initial enrolment population of 1986 and 1987 have completed their attempts at the course.

Conclusions

In a course with a current active enrolment of over 400, which has had over 2000 assignments processed during the last 18 months, and a course with only three part-time tutors, the cost-effectiveness and time-efficiency of running the course as a computer based education package is self-evident. Coupled with the positive response from student evaluation, and the emerging success patterns of individuals who complete the course, it would appear the course is successfully meeting its two main objectives as outlined at the beginning of the paper.

However, there are still important issues in the evaluation of the course which are currently under examination, and need to be finalized before a complete evaluation of the course can be tabled. With the introduction of the interactive micro-computer based assessment in 1988, and with the completed statistics of the 1986 and 1987 student populations, it will be possible to instigate additional evaluation as to the effects on students, and the efficiency of the various modes of assignment processing. More specifically, it will enable insight into issues related to the utilization of computers in the delivery, processing and monitoring of course material on a large scale.

Bibliography

Barker, L.J., White, V.J., and Taylor, J.C. (1984)Computer managed learning in tertiary education: An organisational development perspective. *Australian Journal of Adult Education*, 25(1), 23–30.

Crock, M.J. (1987). Preparatory studiesat the Darling Downs Institute of Higher Educatio: A case study in alternative entry to tertiary education. Paper presented to ASPESA 8th Biennial Forum, University of New England, July.

Michael J. Crock is currently employed as a Lecturer - Instructional Designat the Darling Downs Institute of Advanced Education, where his duties include the design of print, video, audio and computer–based courseware for external and internal courses, and he manages the Computer Managed Learning program at DDIAE. Current teaching involves remedial and tertiary level mathematics. Current research involves, the effective and efficient utilisation of instructional design modelss, and the analysis of Computer Based Education systems with reference to the cost effectiveness and the changing educational requirements for such systems.

APPENDIX A

ASSIGNMENT 1A: MODULE 1 - SPIM UNIT 1

This assignment is to be a computer marked test containing both exact answer and multiple-choice questions You should place your answers on the answer sheet immediately following this assignment.

Answer all questions in the format requested, and please note that if an answer is negative you must remember to include a "–" before the value. If your answer is positive do not place a "+" before the value as it is not required. All answers for this assignment should be in integer and/or fraction form, and if you should obtain an answer which includes both an integer and fraction value, it should be placed on the answer sheet in the following form

"$3\frac{1}{3}$" would be written

$\boxed{3}\ \boxed{\&}\ \boxed{1}\ \boxed{/}\ \boxed{3}$

where "&" is the "and" symbol. If too many boxes are supplied for a particular answer remember to leave the end boxes blank.

Appendix A: (continued on page287)

Question		Mod 1	Mod 2	Mod 3	Mod 4	Mod 5	Mod 6
1. I found the module interesting.	M	1.74	1.80	1.81	1.92	1.79	1.83
	N	167	86	43	27	14	12
	SD	0.47	0.40	0.46	0.62	0.58	0.58
2. The module was easy to follow.	M	1.78	2.32	2.32	2.84	2.71	2.92
	N	176	75	47	25	14	12
	SD	0.73	0.62	0.63	0.75	0.89	1.00
3A. The presentation of the module including: - formative assessment items assisted my learning.	M	1.78	1.89	1.88	2.07	2.07	2.17
	N	160	71	42	27	14	12
	SD	0.50	0.43	0.40	0.68	0.83	0.83
3B. The presentation of the module including: - directions to read assisted my learning.	M	1.77	2.00	2.00	2.33	2.14	2.5
	N	163	73	42	24	14	12
	SD	0.54	0.47	0.49	0.71	0.86	1.00
3C. The presentation of the module including: - Study Book assisted my learning.	M	1.68	1.75	1.83	2.15	1.92	2.33
	N	165	73	41	26	13	12
	SD	0.50	0.52	0.44	0.47	1.00	1.07
3D. The presentation of the module including: - statements of objectives assisted my learning.	M	1.82	1.92	1.95	2.23	2.07	2.08
	N	154	73	39	26	14	12
	SD	0.47	0.43	0.51	0.65	0.73	0 51

Table 1: Results for student evaluation

Question		Mod 1	Mod 2	Mod 3	Mod 4	Mod 5	Mod 6
3E. The presentation of the module including: - telephone contact assisted my learning.	M	2.11	2.00	1.95	1.88	2.12	2.00
	N	72	27	19	13	8	5
	SD	0.74	0.48	0.62	0.49	0.64	1.00
3F. The presentation of the module including: - pretest assisted my learning.	M	1.67	1.74	1.74	1.89	2.14	2.08
	N	160	72	39	27	14	12
	SD	0.57	0.50	0.55	0.50	0.86	0.79
3G. The presentation of the module including: - set text assisted my learning.	M	1.68	1.70	1.74	2.25	2.21	2.00
	N	161	73	39	27	14	12
	SD	0.54	0.54	0.55	0.81	0.80	0.74
4. The assessment item is a fair test of my attainment of objectives in this module.	M	1.77	1.78	1.67	1.92	1.78	1.83
	N	164	77	42	27	14	12
	SD	0.52	0.45	0.47	0.62	0.42	0.39
5. External studies facilities provided by DDIAE were adequate for studying this module.	M	1.81	1.94	1.94	2.05	1.92	1.89
	N	155	68	34	21	13	12
	SD	0.52	0.52	0.42	0.66	0.49	0.78
6. I spent approx. ____ hours on this module.	M	8.03	19.65	27.47	37.48	28.33	28.36
	N	162	68	38	25	12	9
	SD	5.96	10.86	16.73	27.97	23.76	24.54

Table 1: (continued)

QUESTION 1

Which of the following are the prime factors of 180?

(Select the correct option.)

1. $2 \times 2 \times 3 \times 3 \times 5$
2. $4 \times 5 \times 9$
3. 1×180
4. $3 \times 3 \times 4 \times 5$

QUESTION 2

Evaluate $3 \div 3^{1/2}(3^0 \times 3^{3/4})^2 + 3^2$

QUESTION 3

Which of the following is the correct simplification of $\dfrac{a^2b^{-1}c}{ab^2c^3}$?

(Select the correct option.)

1. $\dfrac{ab^3}{c^2}$

2. $\dfrac{a}{b^3c^2}$

3. $\dfrac{ab^{-1}}{c^2}$

4. $\dfrac{a^2b}{c^3}$

QUESTION 4

Solve the equation $4t - 10 = 2t - 9$

QUESTION 5

Evaluate $\left(\frac{1}{3} + 2 \div 6 + \frac{4}{6}\right) \times 2 - \left(\frac{1}{3}\right)^2$

QUESTION 6

Simplify $27^{-1/3}$

QUESTION 7

If $a = \frac{1}{3}$, $b = \frac{2}{5}$ and $c = \frac{3}{5}$ find what $\frac{3ab}{2bc}$ equals.

QUESTION 8

Solve for x: $3x + 7 = x + 11$

QUESTION 9

Solve for t: $\frac{1}{t} + \frac{3}{2t} + 4 = 7 - \frac{1}{2t}$

QUESTION 10

Which of the following is the correct simplification of $x^2y(x^4z + xy^3)$?

(Select the correct option)

1. x^7y^4z
2. $x^{-2}yz + xy^2$
3. $x^6 + yz + x^3 + y^4$
4. $x^6yz + x^3y^4$

COMPUTER MARKED TEST

Place your assignment sticker here	**ANSWER SHEET**
	FOR OFFICE USE ONLY		

TEST 11032 - Assignment 1A	DUE DATE Week 2

Please remember that you must include a "–" if your answer is negative but <u>do not</u> include a " + " if it is positive. If too many boxes are supplied for a particular answer leave the end boxes blank.

1.	☐	(1:21)
2.	☐ ☐ ☐ ☐ ☐	(1:22 - 1:26)
3.	☐	(1:27)
4.	☐ ☐ ☐ ☐ ☐	(1:28 - 1:32)
5.	☐ ☐ ☐ ☐ ☐	(1:33 - 1:37)
6.	☐ ☐ ☐ ☐ ☐	(1:38 - 1:42)
7.	☐ ☐ ☐ ☐ ☐	(1:43 - 1:47)
8.	☐ ☐ ☐ ☐ ☐	(1:48 - 1:52)
9.	☐ ☐ ☐ ☐ ☐	(1:53 - 1:57)
10.	☐	(1.58)

```
***************************************************************************
UNIT : 11032  01   X  : UPDATING MATHEMATICAL SKILLS
ASSESSMENT :     21783 : DIAGNOSTIC QUIZ
STUDENT : D87[redacted]    : MR PETER [redacted]
***************************************************************************


                     MR PETER [redacted]
                     P O BOX [redacted]
                     ROMA
                     QLD
                     4455



-------------------------------------------------------------------------
*
STUDY BOOK:

Page 55 - Equation 7 should read a=10

Page 49 - Answer to Question 1 DOES NOT include "-1, -2, -3, etc.
          and zero" as natural numbers only contain the counting
          numbers: 1, 2, 3, etc.

Page 97 - Reference page in text for 'Factorial n' is page 93 of


TEXT BOOK:

Page 32 - Exercise 2.4, No. 1A: Interval should read: [-3,5)
                            No. 1B: Interval should read: [-1,1]

Page 39 - Point Q co-ordinates should read (-1,1/2)

Page 46 - Formula for "d" following figure 3.13 should finish
          with "(y subscript 2 minus y subscript 1)squared"

Page 49 - Exercise 3.6, No. 3: should read: y=-0.15x

Page 67 - 2nd example at top of page should read: E={2,4,6...}

Page 121- Top of page should read "Exercises 6.6"

Page 148- Figure 7.21 is incorrectly labelled. Function should
          read "inverse function = 1/x," not "x+2"

Page 171- In Figure 8.6, the functions all cross the y-axis at y=1,
          the graph has the position of 1 on the y-axis marked
          incorrectly.

Page 184- Figure 9.6. Hypotenuse should be "square root of 2".

Page 186- Figure 9.11. Point on x-axis should be labelled "-1".

Page 326- 1b Answer should be prefaced with "1/3".
*END
-------------------------------------------------------------------------
```

- a statement as to correctness/validity of the response
- feedback is provided where appropriate

```
----------------------------------------------------------------------
  QUESTION     RESPONSE                                RESULT

     1         +12                                     CORRECT
     2         -7                                      CORRECT
     3         -1                                      CORRECT
     4         +4                                      CORRECT
     5                                                 NOT ATTEMPTED
     6         +125                                    CORRECT
     7                                                 NOT ATTEMPTED
     8                                                 NOT ATTEMPTED
     9                                                 NOT ATTEMPTED
    10                                                 NOT ATTEMPTED
    11         +1/2                                    CORRECT
    12         +29                                     INCORRECT
    13         +7                                      CORRECT
    14         +10                                     CORRECT
    15                                                 NOT ATTEMPTED
    16                                                 NOT ATTEMPTED
    17         +1                                      CORRECT
----------------------------------------------------------------------
```

Question 5

The correct answer is -1/15 because

```
        (-1/5) x (-4/3) - 1/3 = (-1/5 x -4/3) - 1/3
                              = 4/15 - 1/3
                              = 4/15 - 5/15 = -1/15
```

Question 7

a^{-1}/a^{-3} is equal to a^{3}/a^{1} = $a \bullet a \bullet a/a$ = a^{2}

Therefore the final power of "a" = +2.

Question 8

$a^{2}b^{-2}/a^{-3}bc$ = $a^{2}.a^{3}/b.b^{2}c$ = $a^{5}/b^{3}c$ = $a^{5}b^{-3}c^{-1}$

Therefore the final power of "a" = +5.

Question 9

$a^{2}b^{-2}/a^{-3}bc$ = $a^{2}.a^{3}/b.b^{2}c$ = $a^{5}/b^{3}c$ = $a^{5}b^{-3}c^{-1}$

Therefore the final power of "b" = -3.

Question 10

$a^{2}b^{-2}/a^{-3}bc$ = $a^{2}.a^{3}/b.b^{2}c$ = $a^{5}/b^{3}c$ = $a^{5}b^{-3}c^{-1}$

Therefore the final power of "c" = -1.

Question 12

Remembering the correct order of operations, the solution is

$$(16 \div 2^{2} + 3^{2} \times 2) \div 2 + 4^{2} =$$

```
        (4 + 18) ÷ 2 + 16 =
        22 ÷ 2 + 16 =
        11 + 16 = +27
------------------------------------------------------------------------
Question  15


The correct solution is:

        yz/2x = (4/3•2/5)/(2•1/3) = (8/15)/(2/3) =

        8/15 • 3/2 = 24/30 = +4/5
------------------------------------------------------------------------
Question  16

The correct solution is:

        4x + 7 = 5x + 4
       4x - 5x = 4 - 7
           -x = -3
            x = +3
------------------------------------------------------------------------
          Number of questions                   :  17
          Number answered correctly             :  9
          Number answered incorrectly           :  1
          Number with answer missing            :  7
          Number with invalid answers           :  0
          Score  ( % )                          : 52.9
------------------------------------------------------------------------

You have scored between 50% and 70% on the diagnostic quiz which
suggests that while a little rusty, you have a reasonable
understanding of the fundamentals of mathematics.  Please feel free
to contact D.E.C.E. should you have any outstanding queries on the
diagnostic quiz.  If you have no queries you should now begin
studying Module 1.
```

APPENDIX C

```
                              ASSIGN06.PUB.SR   04/AUG/87    19:02   PAGE 49

                DARLING DOWNS INSTITUTE OF ADVANCED EDUCATION

                              UNIT ASSESSMENT REPORT

86 11032  UPDATING MATHEMATICAL SKILLS 01  X     RECD      RETD      RESULT

D86410XXX    MR MAURICE X              (CONFD:  09MAY86)
AS  01  ASSIGNMENT 1A          S                 18JUL86   23JUL86   50/
AS  02  ASSIGNMENT 2A          S                 15SEP86   17SEP86   73/
AS  03  ASSIGNMENT 3A          S                 29SEP86   03OCT86   70/
AS  04  ASSIGNMENT 4A          S                 16OCT86   22OCT86   66/
AS  05  ASSIGNMENT 5A          S                 27NOV86   05DEC86   30/
AS  06  ASSIGNMENT 6A          S                 11DEC86   15DEC86   30/
AS  07  DIAGNOSTIC QUIZ        S                 09MAY86   19MAY86   58/
AS  08  ASSIGNMENT 1B          S                 *O'DUE*
AS  09  ASSIGNMENT 1C          S                 *O'DUE*
AS  10  ASSIGNMENT 2B          S                 *O'DUE*
AS  11  ASSIGNMENT 2C          S                 *O'DUE*
AS  12  ASSIGNMENT 3B          S                 *O'DUE*
AS  13  ASSIGNMENT 3C          S                 *O'DUE*
AS  14  ASSIGNMENT 4B          S                 *O'DUE*
AS  15  ASSIGNMENT 4C          S                 *O'DUE*
AS  16  ASSIGNMENT 5B          S                 21JAN87   23JAN87   100/
AS  17  ASSIGNMENT 5C          S                 *O'DUE*
AS  18  ASSIGNMENT 6B          S                 21JAN87   23JAN87   90/
AS  19  ASSIGNMENT 6C          S                 *O'DUE*

D80601XXX    MR ALLAN J Y              (CONFD: 15APR86)
AS  01  ASSIGNMENT 1A          S                 29APR86   14MAY86   90/
AS  02  ASSIGNMENT 2A          S                 23MAY86   28MAY86   93/
AS  03  ASSIGNMENT 3A          S                 *O'DUE*
AS  04  ASSIGNMENT 4A          S                 *O'DUE*
AS  05  ASSIGNMENT 5A          S                 *O'DUE*
AS  06  ASSIGNMENT 6A          S                 *O'DUE*
AS  07  DIAGNOSTIC QUIZ        S                 22APR86   02MAY86   52/
AS  08  ASSIGNMENT 1B          S                 *O'DUE*
AS  09  ASSIGNMENT 1C          S                 *O'DUE*
AS  10  ASSIGNMENT 2B          S                 *O'DUE*
AS  11  ASSIGNMENT 2C          S                 *O'DUE*
AS  12  ASSIGNMENT 3B          S                 *O'DUE*
AS  13  ASSIGNMENT 3C          S                 *O'DUE*
AS  14  ASSIGNMENT 4B          S                 *O'DUE*
AS  15  ASSIGNMENT 4C          S                 *O'DUE*
AS  16  ASSIGNMENT 5B          S                 *O'DUE*
AS  17  ASSIGNMENT 5C          S                 *O'DUE*
AS  18  ASSIGNMENT 6B          S                 *O'DUE*
AS  19  ASSIGNMENT 6C          S                 *O'DUE*
ASSIGNMENT                   ENRL OUTS SUB RESUM EXT %RESP RETD %RET

ASSIGNMENT 1A                152   77   75  0     0  49.34  75  100
ASSIGNMENT 2A                152  102   50  0     0  32.89  50  100
ASSIGNMENT 3A                152  116   36  0     0  23.68  36  100
ASSIGNMENT 4A                152  134   18  0     0  11.84  18  100
ASSIGNMENT 5A                152  135   17  0     0  11.18  17  100
ASSIGNMENT 6A                152  135   17  0     0  11.18  17  100
DIAGNOSTIC QUIZ              152   52  100  2     0  65.78 102  100
ASSIGNMENT 1B                152  141   11  0     0   7.23  11  100
ASSIGNMENT 1C                152  142   10  0     0   6.57  10  100
ASSIGNMENT 2B                152  147    5  0     0   3.28   5  100
ASSIGNMENT 2C                152  147    5  0     0   3.28   5  100
ASSIGNMENT 3B                152  149    3  0     0   1.97   3  100
ASSIGNMENT 3C                152  150    2  0     0   1.31   2  100
ASSIGNMENT 4B                152  150    2  0     0   1 31   2  100
ASSIGNMENT 4C                152  151    1  0     0   0.65   1  100
ASSIGNMENT 5B                152  150    2  0     0   1.31   2  100
ASSIGNMENT 5C                152  151    1  0     0   0.65   1  100
ASSIGNMENT 6B                152  149    3  0     0   1.97   3  100
ASSIGNMENT 6C                152  151    1  0      0  0.65   1  100

NO. *DROPPED:  42
```

APPENDIX D

ATTENTION Michael Crock, D.E.C.E.
OPINIONNAIRE Course 11032 Module 1

Please return this to the Administrator, D.E.C.E., with Assignment 1.

Tick the box which best describes your reponse to the statements made and add any comments you wish to make. Further specific comments on how the module/course could be improved are welcomed and should be added overleaf or on attached sheets.

	strongly agree	agree	disagree	strongly disagree	
1. I found the module interesting.	☐ 1	☐ 2	☐ 3	☐ 4	(1.10)
Comments:					
2. The module was easy to follow.	☐ 1	☐ 2	☐ 3	☐ 4	(1.11)
Comments:					
3. The presentation of the module including:					
- formative assessment items } assisted my understanding	☐ 1	☐ 2	☐ 3	☐ 4	(1.12)
- directions to read	☐ 1	☐ 2	☐ 3	☐ 4	(1.13)
- Study Book	☐ 1	☐ 2	☐ 3	☐ 4	(1.14)
- statements of objectives	☐ 1	☐ 2	☐ 3	☐ 4	(1.15)
- telephone contact	☐ 1	☐ 2	☐ 3	☐ 4	(1.16)
- pretest	☐ 1	☐ 2	☐ 3	☐ 4	(1.17)
- set text	☐ 1	☐ 2	☐ 3	☐ 4	(1.18)
Comments:					
4. The assessment item is a fair test of my attainment of objectives in this Module.	☐ 1	☐ 2	☐ 3	☐ 4	(1.19)
Comments:					

please continue over

	strongly agree	agree	disagree	strongly disagree	
5. External studies facilities provided by DDIAE were adequate for studying this module.	☐ 1	☐ 2	☐ 3	☐ 4	(1:20)

Comments: ....

....

6. I spent approximately ☐☐ hour on this module.

Comments: ....

....

7. I would improve the module by

....

....

Thank you

Michael Crock

Are computers the write stuff? Computers and writing instruction

Marsha Durham, School of Humanities and Applied Social Sciences, Nepean College of Advanced Education

A variety of computerized writing aids exist, eg for topic exploration, spelling and grammar checking, style analysis and word processing. Educators may consider the aids unproblematic and even desirable in composition instruction, especially as students' enthusiasm for them seems to ensure their success.

The effect of these 'writer's helpers' on students' writing has not been thoroughly studied. Preliminary studies suggest that the computer may be of limited value. Writing is both individual and recursive, and programs based on a superficial or prescriptive view of writing cannot duplicate the complex choices that writing involves.

To effectively use computers in writing instruction, educators must be prepared to act as mediators between students and technology, by (1) helping students move from the limitations of these programs to expand their own choices in writing, and (2) working with technical experts to create programs that incorporate sound educational objectives.

Two major problems have weakened much of the research about the effect of computers on students' writing: conclusions based on insufficient or biased data; and reliance on anecdotal evidence as 'proof'. In addition, the overly enthusiastic tone of many articles about computers and writing promotes uncritical acceptance:

> You've written several late passes ... for students too involved in their work to leave ... (Y)ou lock your door against a roomful of students working after school. You arrive home only to receive phone calls from students too interested in their assignment to wait until class ... Experiencing such rampant student enthusiasm is a rare and isolated phenomenon for an English teacher. It is however, a commonplace experience for a computer teacher! (Leonardi and McDonald,1987, p. 45)

The computer's abilities for writing assistance are appealing, as it can provide layouts, previews, spelling checks, style analysis and grammar instruction, as well as perfect presentation. Yet, as with any technology introduced for educational purposes, computerized aids for writing should be evaluated on their ability to support pedagogical objectives. Many problems surround the technology, with some programs presenting inappropriate procedures and criteria as part of their writing instruction. In this paper, I will focus on the impact that writing programs have on student writers, and comment on their effect in changing students' writing processes and their interactions within the writing class.

Views of the writing process

Although often taught as separate stages, the major cognitive processes in writing — planning, translating, reviewing and monitoring — can be implemented at any point in one's writing (Sommers, 1985, p. 5). Writing is recursive, with writers repeating their processes from first idea through to finished product. For example, although editing is often thought of as a final action, it also happens before and during one's writing.

Writing is situationally based, dependent on such variables as one's interest and knowledge of a topic, perception of audience, past successes in writing, subject limitations and time constraints. As well, writers have different, individual writing styles. 'Mozartians', for example, make extensive outlines, mentally or on paper, before writing a comprehensive first-final draft. Another type, the 'Beethovians', favor writing a quick, rough draft, then revising extensively, perhaps arriving at a clear pattern only late in the process.

Recent composition research (Berthoff, 1981; Stock, 1983; Beach & Bridwell, 1984; Flower, 1985; Knoblauch & Brannon, 1984; Tate & Corbett, 1981, Hillocks, Jr, 1986) supports the view that writing is more than just exhibiting one's learning. It is a way of learning, for our meaning evolves as we make choices in our writing. Unfortunately, choices diminish when students write, as they manufacture 'Engfish' (see Reference Note 1), ie lifeless writing which has lost its ability to 'speak' to readers. 'Engfish' results when students are are unable or uninterested in ensuring that their meaning is reciprocated by the reader.

Can computers play a constructive role in teaching students to write for meaning? Most computer writing programs offer only limited assistance in teaching the complex choices that good writing involves. While useful for specific purposes, these programs can be damaging if they are introduced as comprehensive tools, with students expected to curtail their writing to use the program.

Computerized heuristics

Academia's emphasis on evaluation, style, grammar and punctuation may cause students to 'freeze' in their thinking, and explore only the safest ideas. Instead, students need to be encouraged to experiment, in order to discover what they want to write about, what they know about the subject, and what they must research:

> The first use of language that a student of composition has to learn...is in the generation of chaos .If we don't begin there, we falsify the composing process because composition requires choosing all along the way, and you can't choose if there are no perceived alternatives: chaos is the source of alternatives. If we are unwilling to risk chaos, we won't have provided our students with the opportunity to discover that ambiguities are, as I.A. Richards has said, 'the hinges of thought'. (Berthoff,1981: 75)

Too often students curtail thinking too early, neglecting to generate choices, eg ideas, questions, inconsistencies, problems and qualifications. Researchers believe that one major difference between good and poor writers is their ability to possess and select different heuristics: "Good writers not only have a large repertory of powerful strategies, but they have sufficient self-awareness of their own process to draw on these alternative techniques as they need them" (Flower, 1985, p. 37).

Computers can help in teaching heuristics. Programs have been written for the most popular types, eg freewriting, brainstorming, tagmemics, treeing, Aristotle's topoi, Burke's dramatistic pentad, and discovery questioning, as well as for specialized writing topics, eg narration, argumentation, poetics and fictional characterization.

Computerized heuristics can help students who are unsure about their ideas, and who might not speak up in a class dicussion about topics. The dialogue format of some heuristic programs encourages students to go beyond their commonsense understanding to consider different aspects of a topic, and amass more significant and thoughtful information. The program can also point out where they lack sufficent knowledge. Students introduced to a wide repertory of invention strategies may learn to be more flexible in choosing ways to explore ideas according to a particular writing task. Brainstorming, for example, may work for a short creative writing piece, while a different heuristic may be more suitable for a lengthy research report.

Using these programs can create problems. If only one heuristic is available, students may have the mistaken view that it is relevant for every writing task. Students may become overly dependent on software, and wait passively for cueing. Some programs may be so complicated that students cannot relate them to their assignment. Others may alienate students by using a lock-step approach, or an inappropriate tone of response. And, on their own, heuristics programs will not overcome students' writing apprehension or their preoccupation with 'regurgitating' the lecturer's ideas. By undermining students' interest in human feedback, programs may isolate students from perceiving their writing as a meaningful, communicative act for others.

Educators can thoughtfully integrate a heuristics program into the writing class by pointing out its immediate and long-term advantages and disadvantages, or by comparing responses of the program with those of a real audience. They can ensure that students do not rely on the program for all their writing, but incorporate responses from others, as well as think up their own questions in considering their topic. This critical approach allows students to understand the program's limitations, and the value of knowing and using a variety of methods in assessing a topic.

Impact of word processing

Researchers interested in the writing process have usually relied on the subjective method of having writers describe their composing habits, either as they write, or in retrospect. Studying students' modes of writing is even more difficult because of the negative influence of having topics assigned, and the finished work evaluated. Despite breakthroughs in research (see Reference Note 2), not enough is known about writing processes to adequately evaluate the impact of computer use.

The hardware and software is so often seen as beneficial, that it is difficult to consider that both can have a detrimental effect on students' writing. For example, word processing offers 'clean copy' by immediately incorporating changes. Seeing their first draft perfectly displayed can give students a false sense of its importance, and thus undermine the value they perceive in reflecting on their work, and editing it. Not being able to refer to their original and subsequent drafts may rob students of a sense of achievement. Resurrecting discarded information is impossible with some programs, yet this is often necessary when the writing takes a new direction.

Computer hardware can also inhibit student's writing. A mouse, for example, may still be more cumbersome than a pencil, and therefore editing with it is avoided. The computer screen, which allows only a limited view of one's work, makes it difficult to assess overall movement. Commands like 'scroll' and 'find', and features like split screens and clipboards,are helpful, but these still do not allow the simultaneous viewing of major sections that some writers may need. The screen may prevent interaction, as students stayed 'glued' to their work. And, it can inhibit trials and other types of experimenting, because it creates a 'public' display of one's work. Even seeing one's work in an electronic format can be alienating, as one student found: "It's awfully hard for me to look at the screen and want to edit anything. My paper seems so far away, almost abstract" (Harris, 1985, p. 327).

Absence of definitive knowledge about computers' influence on writing has not prevented some educators from assuming that students who use word processing not only write more, but edit more significantly. Preliminary research, however, suggest that students do not fully utilize the editing functions in word processing. A recent study indicated that although observed students believed that they revised more on the computer, they did not. In addition, changes made were mainly minor, eg "adding a few details, rearranging words within a sentence, substituting one word for another of essentially the same meaning, deleting a word or phrase, etc" (Harris, 1985, p. 328).

In surface revision, the computer is only a 'glorified typewriter', used to transcribe handwritten work, correct surface errors, and make attractive pages. Significant revision, on the other hand, includes making "semantic and rhetorical changes that affect the content and organization of a piece of discourse" (Harris, 1985, p.323). Writers revise significantly when they concern themselves with making their prose reader-based : they actively test their meanings and gauge their readers' comprehension. Students who are writer-based believe that it is sufficient if their writing is clear to them, unaware that meanings are not so easy to establish:

> The making of meaning is a dialectical process determined by perspective and context. Meanings change as we think about them; statement and events, significance and interpretations can mean different things to different people at different times. Meanings are not prebaked or set for all time; they are created, found, formed, and reformed. (Berthoff, 1981, p. 71)

In creating meaning, the computer can be useful, as it allows experimentation: ideas can be developed, erased, or put on 'hold'. Once students view the computer as a good 'trialler', that can easily clean up false starts, weak writing, and undeveloped sentences, they may move naturally to significant revision.

Text graders

A full-time composition instructor in an American college writing program may mark over 350 assignments in a 10 -week term. Consequently, interest has been keen in computerized text grading, which allows educators to flag common mistakes and leave the program to provide detailed explanations to students.

What is the pedagogical worth of this software? Educators may decide that the computerized grader provides extensive information while freeing them to concentrate on global comments. An instructor can mark 'CS' on an assignment, for example, and leave the program to define 'comma splice' to the student, and suggest exercises related to this error.

The drawback to the program's ability to pinpoint errors is that it may encourage lecturers to flag every mistake, in content, style, grammar, spelling and punctuation. Not only do they then spend more time marking, but such intensive correction may be counter-productive. Students may see the lecturer's comments as little more than "mutilations of their work and intrusions into their ideas", and believe that the comments show that "they haven't expressed the approved opinions" (Sudol, 1985: 333).

In addition, students often have little understanding that the lecturer's comments are to aid them in their future work. Consequently, composition researchers are becoming interested in the value of checking work in progress, rather than finished pieces. Instead of reading for mistakes, the lecturer's interest must necessarily shift to searching for intended meaning, attempting "...the dialogue that has been formed in the writer's mind, transformed in his dialogue on paper" (Berthoff, 1981:39). Surface errors are ignored until the student learns how his or her intended meaning has been understood.

One objection to this idea is the amount of time taken to read several drafts. Proponents suggest, however, that intervening earlier in the writing process saves one from reading lifeless, inaccurate essays. Students can be taught how to give valuable comments on others' work, and their analysis can be followed by computerized checks for spelling and grammar. The final product, accurate and edited, allows the lecturer to read it without having to combine the role of critic with that of proofreader and editor.

Checkers

Software exists that can check text for spelling, grammar, punctuation, word usage, and style. The expectation is that the information produced will help writers understand their weak points in a work, and revise. In advanced or specialized writing classes, eg technical and professional writing, this information can be valuable, establishing standards of 'correct' writing. For general writers, it may be detrimental unless students understand the program's limitations.

For example, spelling programs, by freeing poor spellers from having to concentrate on mistakes, create new interest in writing. Most spelling checkers, however, have vocabulary lists too limited for tertiary level writing, and these would need to be augmented for students' use.

As grammar and punctuation is complex and inextricably related to context, checkers for these elements are usually inappropriate, because they rely on simplistic, prescriptive statements or rules. In one program for younger children, for example, the check consists of asking the writer the same general question for each sentence:"Are you using punctuation correctly?", "Is this a good sentence?"(Smithson, 1987: 88). Students needing the check program probably would not be able to answer these questions.

Style checkers pit one's written work against a prescribed readability formula. Researchers have already questioned the worth of using formulae,originally created to check the reading level of textbooks, as a standard for evaluating other types of writing. Readability formulae suggest that word length, sentence length, abstractions, and other stylistic features constitute the work's level of comprehension, and therefore its worth. These features are inherent features of writing, but it does not follow that obeying the program's suggestions will automatically produce good writing.

While readability formulae may be useful to professional writers, students may find that the analysis of their writing is presented in an almost indigestible format. They must be taught how to translate the evaluation into specific strategies for improving their writing, if the program is to be useful.

Future development of writing programs

Students need direction in understanding what may be wrong with their writing. But the choices that they make are within the context of each specific rhetorical situation for which they write. The program that can take account of all the variables in those situations has yet to be created.

In the future, valuable programs for writing, would be sophisticated packages incorporating all 'stages' of the writing process. For example, students could concentrate on shifting from a writer-based to a reader-based view as they revise if they could use a simple heuristic/word processing program that asked them to answer questions about their topic, enter their first draft, then answer further questions about how they had developed their ideas. Such comprehensive writing programs have already being developed, but are still limited for full classroom use.

"Engfish" may stem from students opting for the first, easy response to a topic. If students could select from a variety of heuristics on one disk, they might be encouraged to explore not only more possibilities within the topic, but also learn what heuristic style most suits them. Another helpful program would be one that assists students in selecting their own topic, based on analyzing their knowledge, interests and abilities.

Graders and checkers are reasonable inclusions in writing instruction, if they include further information and exercises for students who wish to work on specific writing problems. If these 'help' items incorporated longer passages of text, not just isolated sentences, students would learn how to spot problems in 'real-life' prose that they read. Drill-practice programs in grammar, punctuation, and spelling have lost their popularity, but they may offer a non-threatening way of providing encouragement and revision of basic writing skills to particular types of students, eg mature-age students enrolling after a gap in their formal education..

Conclusion

Educators interested in computers for writing should be aware of more than the technical aspects. The cognitive and psychological aspects of writing lead educators to question the worth of 'writers' helpers' that use out-moded prescriptive models, are overly difficult to use, encourage a 'piecemeal' approach to writing, or focus too stringently on errors.

Before any computerized writing program is introduced, the technology should be evaluated to see what benefits it offers to students' writing and to ensure that inappropriate learning experiences are not being designed to fit the software. Instead, software that teaches students to expand their writing choices, and to understand and explore their methods, will allow students to see technology as a help, rather than another hurdie in what may too often be seen as the 'mystery' of good writing.

Educators can also assess their ability to mediate between the program's operating methods and students' educational needs. Integrating programs with the course's educational objectives, includes considering how the technology will affect educators' teaching methods, classroom interactions, and other commitments. More writing educators are writing their own computer programs, basing them on the growing body of knowledge and research about the writing process. As computers become more sophisticated, perhaps writing specialists and technical experts will collaborate to develop advanced, comprehensive programs that can accept numerous options and choices, and thus allow students to be truly supported by computers as they experiment in creating reciprocated meanings in their writing.

Appendix A: Computer Programs for Writing

Prewriting

1. QUILL Planner
Apple
Prompts by keywords, title, author, entry numbers, topic

2. Narcissus & Echo authoring system
Reads list of questions into an invention program

3. SEEN
Apple
Prompts for fictional character analysis

4. TOPOI
DEC, Apple, IBM PCPrompts for purposes regarding subject topic

5. Prewrite
Apple
Prompts for fictional/personal prose, or expository prose, then suggests freewriting

6. Interaction
Bulletin board programs for posing questions to class or creating database of questions.

Comprehensive; Specific

1. Writing Workshop
Apple
Grades 3-10
includes word processor, Prewriting (brainstorming, branching, nutshelling). Postwriting (checks mechanics, spelling, and proofreads)

2. Poetics
IBM
Tutorials, exercises and authoring system for teaching prosody at tertiary level

3. The Poetry Processor: Orpheus A-B-C
(not available as of Feb 1986)

4. WARP
Includes drafter, editor, outliner, printer, utilities
(being developed)

5. Writer's Helper
programs including creation , analysis (readability), revision

Revising and Text analysis

1. ALEXIS
(Dutch program)
error description and correction at 5 levels

2. Catch
Apple
Grammar and style checking, plus prompts for revising

3. CRITIQUE (formerly E.'ISTLE)
IBM 370
Detects 14 classes of syntactic errors in sentences, eg 'who' for 'whom', subject-verb disagreement, improper verb form. Directed to office correspondence

4. Grammatik
CP/M, IBM PC, TRS-80
Prompts for errors in capitalization and punctuation, identifies sexist language, and trite or wordy phrases, and gives sentence and word length.
Includes visual aids /spreadsheets/ printouts possibly used by technical writing students for reports.
Has audible on-going spelling check.

5. Homer
Apple
Isolates prepositions, 'to be' verbs and imprecise words, and gives sentence lengths and word types.

6. Punctuation & Style
CP/M, MS-DOS
Checks punctuation errors, unpaired format commands, double words, passive voice, misused/overworked phrases

7. Style
Includes Bias, which calculates ratio of female to male pronouns, plus flags sexists words and phrases.

8. Unnamed
Prof. B Broughton & W Horn, Clarkson University

program that flags libelous and slanderous statements.

9. Styled, Styllist
IBM
Checks long words, punctuation, 'structure' words, forms of 'to be', norminalizations.

10. Writer's Workbench
UNIX
programs: Proofreading, stylistic check, checks for split infinitives, spelling and punctuation errors, overly long sentences, passives, word phrases; readability.

Readability

1. Readability Analysis
Apple
Runs readability tests: Spache, Dale-Chall, Fry, Raygor, Flesch, Gunning-Fog

2. Readability
Apple
Spache, Dale-Chall, Fry Raygor, Flesch, Gunning-Fog, Smog
Can make on-screen changes through editor function

3. CRES (Computer Readability Editing System)
Developed by US Navy to access technical and instructional material. Flags uncommon words, misspellings, long sentences, passive sentences, 'to be' constructions and gives readability score. Specialized word lists.

Drill and practice

eg Text Critique, RSVP, Camelot, Writer/Grader/Reader; Report

Reference Notes

1. The term was coined by the well-known American writing educator, Ken Macrorie.
2. Researchers are using computers to record every keystroke that students make as they write. The resulting information is then linked with a new type of retrospective protocol analysis, in which students describe their writing choices as they watch their changing drafts on video. Details of this research is found in Hawisher, 1986.

Bibliography

Beach, R. & Bridwell, L. (Eds.) (1984) *New directions in composition research.* New York: The Guilford Press.

Berthoff, A. (1981). *The making of meaning: Metaphors, models and maxims for writing teachers.* Upper Montclair, NJ: Boynton/Cook.

Bridwell, L.S., et al. (1984). The writing process and the writing machine: Current research on word processor relevant to the teaching of composition. In R. Beach & L. S. Bridwell (Eds.), *New directions in composition research* New York, Guilford Press, pp 381-398.

Flower, L. (1985). *Problem-solving strategies for writing.* San Diego, CA: Harcourt, Brace, Jovanovich.

Harris, J. (1985). Student writers and word processing: A preliminary evaluation. *College Composition and Communication, 36*(3), 323-330.

Hawisher, G.E. (1986) Studies in word processing. *Computers & Composition,* 4(1), 6-31.

Knoblauch, C.H. & Brannon, L. (1984) *Rhetorical traditions and the teaching of writing.* Upper Montclair, NJ: Boynton/Cook.

Leonardi, E.B., & McDonald, J.L. (February 1987). The micro in the English classroom: Shifting the emphasis from processing words to processing ideas. *Educational Technology,* 45-47.

Smithson, I. (1987). The writing workshop. *Computers and Composition,* 4(1), 78-94.

Sommers, E. (1985). Integrating composing and computing. In J.L. Collins & E.A.

Sommers (Eds.), *Writing on-line: Using computers in the teaching of writing.* Upper Montclair, NJ: Boynton/Cook.

Stock, P., (Ed.) (1983) *Fform: Essays on theory and practice in the teaching of writing.* Upper Montclair, NJ: Boynton/Cook.

Sudol, R. (1985). Applied word processing: Notes on authority, responsibility and revision in a workshop model. *College Composition and Communication, 36*(3), 331-335.

Tate, G. & Corbett, E. (Eds) (1981) *The writing teacher's sourcebook.* New York: Oxford University Press.

Marsha Durham lectures in Communication, specializing in written communication and social interaction (interpersonal, group, organizational). She is interested in different computer uses relevant within the BA (Applied Communication Studies) course at Nepean CAE: CAI for written argumentation and revision strategies in writing, desktop publishing, computer conferencing and word processing. Presently, She is helping to develop a program to teach students how to locate and evaluate fallacies of reasoning.

Assessment of student computer programming assignments

Helen Hasan
School of Industrial and Administrative. Studies
University of Wollongong

Considering the large numbers of university computing students and the scarcity of teaching staff, too many tutor-hours are tied up manually marking student programming assignments. The possibility exists for much of the assessment to be automated since the student programs are already in a computer readable form. Many departments teaching computing do have some on-line systems whereby student programs are collected and some checking done. However in most cases the tutor is at least required to check the source code for such aspects as structure, readability and adherence to specifications. This paper discusses the requirements for an automated system that carries out much of the tutor's task. It describes the performance of such a system used for a course which teaches FORTRAN to second year Diploma students at this university.

The teaching of computer programmming at tertiary level continues to involve large numbers of staff and students in a variety of departments. The assessment of student programs is frequently carried out manually by tutorial staff. This process tends to be unreliable due to its subjectivity and well as occupying tutor time that could better be spent elsewhere. This paper discusses a system which has automated this process for a course which teaches FORTRAN to second year Diploma students at this university.

With computer assessment of non-computing subjects there is often a problem of combining the expertise in the subject matter with the ability to write the required computer package. With a computer programming subject this should present no problems as the teaching staff should be able to design and write an assessment package themselves. Another bonus for this type of system is that the students' work is already on the system in a computer file.

Indeed many departments teaching computer programming do have some on-line system whereby student programs are collected and some checking done, for example Bailes 1983, Whale 1983, Hext & Winings 1969. However this is usually limited to a run of the program against test data. In most cases the tutor is still required to assess the source code for such aspects as structure, readability and adherence to specifications.

The tutor's assessment of a computer program is in some respects akin to the assessment of an English essay with corresponding complexities. In some aspects it would be almost impossible to replace human judgement with a computerized process, for example in assessing the relevance of comments Although this is beyond the capabilities of the type of system used here, in general the prospect of objectivity and consistency, together with the saving of valuable tutor time should tip the balance in favour of an automated system.

The System Design.

The first task in designing this system was to decide what criteria should be used in evaluating the students' work. It was necessary to look at the way tutors were currently operating as well as how the assessment would reflect the overall objectives of the course. It should be kept in mind that the assignments were given to the students for both teaching and assessment purposes.

For the FORTRAN course involved the following categories emerged as the main areas for assessment:

a. Adherence to standard language syntax
b. Use of appropriate data structures
c. Adherence to accepted code structure (sequence, selection and iteration)
d. Appropriate modular programming
e. Readability and documentation
f. Adherence to the assignment specifications
g. Validation of the program with test data.

The next step was to identify what empirical measurements could be made in the categories listed above. Compiling the program would help test category (a) while running the program with test data could be used for category (g). It was not difficult to implement procedures to first of all collect the students' source code, and then test compile, link and run them.

Categories (b) to (f) would on the other hand require some analysis of the the text of the source code. For this a substantial program was written to examine the student source code, producing both a report for the student and a mark which, combined with a mark for the compilation and run, gave an overall grade for the assignment. An overview of this program now follows.

Overview of source code assessment program.

Without making a full semantic analysis it was necessary for the program to recognise FORTRAN statements and some features within them. Given the student source code, first the modular structure and then the code structure within each module is identified. Several checks are then made. Indentation and the use of comments are used as a measure of readability. All variables must be declared, suitably assigned and used appropriately as arguments to subroutines, in common statements or as arrays. Counts are made of all types of FORTRAN statements use, each type of data structure, module etc.

For each assignment the person supervising the marking must decide on numerical constraints for the parameters to be used in the assessment: for example the number of functions and subroutines, the number of 1, 2 etc dimensional arrays, the number of each type of statement and so on. The marker enters these numbers as allowable ranges which are used in the final phase of the system when the mark is generated. These ranges should be decided from the assignment specifications and a look at a few of the students solutions. For example an assignment, where a student is asked to read names into an array from a file using a dummy end of file marker, should have at least one while loop, a 1 dimensional array and so on. It helped if the assignment specifications were designed with the assessment method in mind.

The complete system.

The complete system consists of 5 stages each of which is processed in batch fashion.

The first stage is the collection process. As the students work on stand alone PC's, the tutor sets up a floppy disk running a program which copies the students' source code onto the tutor's disk.

The second and third stages involve batch compilation and run processes with records kept for each student on the success or otherwise at each stage.

The fourth stage is a run of the source code assessment program discussed above.

In the final stage, the output files from the last three stages are merged to produce a report and mark for each student. The students are issued with this computer generated report which includes a summary of their modular and code structure, their use of variables, any marks lost for bad structure, bad indentation, lack of comments, undeclared variables and so on. An example of such a report is shown in Appendix A.

Evaluation of the System Performance.

During its development in 1986, the system was used to assess 5 programming assignments given to 30 to 40 students throughout the course. The system was evaluated in the following three areas:

a. The students involved were surveyed for their opinions of the system.
b. A comparison was made on one assignment of manual marking versus the computer generated mark.
c. A comparison was made of one feature of the students' code (indentation) as the system was used.

The results of these evaluations are as follows:

a. The students were asked 3 questions at the end of the session in which the subject was taught.

1. On the preferred method of submitting programming assignments 70% of students said they preferred the new system of collecting source code to the old method of getting printout of source code and sample runs.

2. On the helpfulness of the computer generated report 10% said it was very helpful, 90% fairly helpful and 0% said it was not very helpful.

3. On the preferred method of assessment 50% said they preferred the computer generated assessment, 25% said they preferred a manual assessment by the tutor and 25% had no preference either way.

b. Assignment 5 was marked both manually and by the computerized system. The resultant marks out of 10 are listed in Appendix B.

A comparison of the two sets of marks show the following:

Average manual mark = 7.75
Average computer mark = 7.95
Average difference (between marks for each student) = 0.5 (6.5% of marks)
75% of each student's marks were within 0.5
90% were within 1.0

The only significant discrepancy in the two marks for any given student was for the very weak students and to a lesser extent for the best 2 or 3 students. For the weaker students the computer generated mark was consistently more generous whereas for the top students it was slightly lower. For the main bulk of students, the agreement between the two marks was within the 1 mark resolution of the marking scheme. It may be significant that the automatic marking scheme tends to be somewhat more conservative than the human marker.

c. A comparison was made on the occurrence of indentation errors in some of the latter assignments compared to the first, before which the students had had no feedback from the system. In the first assignment the students' programs had 4.7% of total lines of code with bad indentation. This dropped down to 1.5% for assignment 3, the largest of their assignments, and remained under 2.0% for the remaining assignments. While there is no comparison with comparable improvement that may have occurred had students' work been marked manually, it seems likely that students tend to take more notice of computer generated diagnostics than they do of the tutor's red pen.

Conclusion.

On the whole the experience with the system in its first year of operation appeared to be beneficial both for the tutor and the students and it was successfully used for the same course again this year (1987).

The system was developed specifically for a course of second year students studying FORTRAN. The number of students was therefore not large and the students were reasonably competent. The advantages of the system in saving of tutors' time and uniformity of assessment would be most valuable in courses involving large numbers of students and many tutors. A system is at present being developed for a first year COBOL course involving 350 students and 12 tutors.

Both the FORTAN and COBOL courses are undertaken mainly by commerce students and so the design of the courses and hence the assessment criteria are in accord with this. It is doubtful therefore whether the systems would translate directly into courses in other departments, say for example Computing Science. However the general principle and design could readily be applied to a wide variety of courses and from the experienced already gained I would suggest that it is an option well worth considering.

Bibliography

Bailes, C.S. (1983) *A Tutorial Introduction to the Submit System*, Wollongong: Department of Computing Science. Wollongong University.

Clarke, R.M. et al (1980). Undergraduate Performance in an Innovative Medical Curriculum Indicators of Performance. *SRHE Proceedings*.

Hasan, H.M. (1986) Automatic Assessment of Student Programming Assignments for the Course AICA 201 User and System Documentation. Wollongong University.

Hall, H.M. and Kidman, B.P. (1975). Empirical Analysis of BASIC and FORTRAN Programs. In O. Lecarme and R. Lewis (Eds).*Computers in Education..*

Hext, J.B. and Winings, J.W. (1969). An Automatic Grading Scheme for Simple Programming Exercises, *CACM*, 12(5).

Hille, R.F. and Higginbottom, T.F., (1983). A System for Visible Execution of Pascal Programs", *Australian Computer Journal* ,15(2).

Microsoft (1984). *MS-FORTRAN User Guide and Reference Manual.*

Sperry (1984). *Guide to DOS 2.11.*

Su, Y.W. and Emam, A.E. (1975). Teaching Software Systems on a Microcomputer: A CAI Approach. In O. Lecarme and R. Lewis (Eds).*Computers in Education..*

Whale, G. (1983). *The Give Report*, Kensington: School of Electrical Engineering and Computer Science, UNSW.

Whitlock, L.R. (1976). *Interactive Test Construction and Administration in the Generative Exam System.* Doctoral Thesis University of Illinois.

Appendix A: Sample Report to Students.

```
AICA 201  PROGRAMMING EXERCISE 4
Student number 11 - GILLON
Compilation in 57.01 secs with 0
error(s).
Run in 30.48 sec.
>> module ASS4
++ module header
++ 3D variable used.
structure:
READ -  CALL SUB -  CALL SUB -
variables used:
```

```
RESULTS(4,4,2): type: integer,
IX: type: integer,
>>> module GAME
structure:
DO - DO - IF - DO - ENDDO -
ENDDO - ENDDO -
variables used:
IX: type: integer, argument.
RESULTS(4,4,2): type: integer, argu-
ment.
I: type: integer,
J: type: integer,
K: type: integer,
>>> module RAND
— value not assigned to function.
structure:
variables used:
IX: type: integer, argument.
>>> module TALLY
structure:
DO - ENDDO - DO - DO - IF - IF -
ENDDO - ENDDO - DO - ENDDO - DO -
IF - ENDDO -
variables used:
RESULTS(4,4,2): type: integer, argu-
ment.
TEAM(4): type: integer,
A(4): type: integer, value assigned,
X: type: integer,
Z: type: integer,
I: type: integer,
J: type: integer,
K: type: integer,
TOP: type: integer, value assigned,
NUM: type: integer, not declared
You have 0 indentation error(s).
Counts: READ 1, WRITE 8, DO 8, IF 4,
WHILE 0  OPEN 2 CALL 2, COMMON 0
Lines of code: 76, comments 32
Modular structure:
program:      ASS4 ->subroutine:
GAME
->subroutine:     TALLY
->real function:  RAND
*MARKS:
Design: 2.00  Program:  4.50  Re-
sults: 3.00  TOTAL:   9.5
```

Appendix B: Comparison of Manual and Computer Generated Marks for Assignment 5

NAME	Manual Mark /10	Computer Mark /10	Difference
JARMAN	3.0	5.2	+ 2.2
JEBARA	5.0	6.8	+ 1.8
ALVAREZ	6.0	6.5	+ 0.5
BOWEN	6.0	6.8	+ 0.8
HEILER	6.0	6.0	0.0
HADIPRODJO	6.5	8.2	+ 1.7
PANIZZUTTI	6.5	7.5	+ 1.0
GURIEFF	7.0	7.2	+ 0.2
GABIN	7.0	6.5	- 0.5
LEE	7.0	8.0	+ 1.0
HAVAN	7.5	7.5	0.0
MARIC	8.0	7.7	- 0.3
ZIPPARO	8.0	8.2	+ 0.2
WACHNIK	8.0	8.5	+ 0.5
VILLARROEL	8.0	8.0	0.0
BISKITZDIS	8.0	7.7	- 0.3
PAINTER	8.0	8.2	+ 0.2
JASIC	8.5	8.5	0.0
CHHOR	8.5	8.0	- 0.5
AYDOGAN	8.5	8.2	- 0.3
MURRAY	8.5	8.5	0.0
CHAND	9.0	8.3	- 0.7
HIDAYAT	9.5	9.7	+ 0.2
SJAMSUDIN	9.5	9.0	- 0.5
TRINH	9.5	9.5	0.0
IVERS	9.5	9.3	- 0.2
DARMOKO	10.0	9.7	- 0.3
DONELLY	10.0	9.5	- 0.5

Readability and documentation

My current position is as a Lecturer in the Information Systems section of the above school as part of the Faculty of Commerce. My background however is in Physics with a BSc and MSc. I recently completed a post-graduate diploma in Computing Science. My teaching interests at present are mainly in computer programming and my personal research interests evolve around the better use of computers in assessment of student computer programs. I am also part of a research group concerned with the effective use of computer technology to applied computer systems training at the tertiary level.

Easing the load of laboratory assessment

Phill Higgins
Applied Science
Gippsland Institute of Advanced Education

At the Gippsland Institute of Advanced Education (GIAE) the drudgery of repetitive marking of laboratory reports has been eased by the introduction of computer assessed laboratory activity. This has been achieved by restructuring the laboratory activity and literature to be compatible with the mainframe computer. The entire student activity is monitored, assessed and recorded without the lecturer/ demonstrator handling one sheet of paper.

The lectures are fairly straight forward — Do some reading, precis a bit here and there, copy a couple of diagrams onto OHP's, check last years notes, work through the examples to make sure they give realistic answers, check how you finished the last lecture and how and where you want to start the next and that is a couple of hours work for a one hour lecture..

Next, the laboratory work — Students already have a set of lab. notes in a booklet, so just a final check on the equipment and a few notes on instructions and announcements and that's it — 15 minutes — easy!!

Now, mark the labs! That's 120 lab. reports at 5 minutes or so each — 600 or more long, dreary, mind-sapping minutes! 10 hours of largely unproductive work! Why can't we afford a tutor/ demonstrator/ external marker? Anything to ease this slogging part of the job is worth a try!.

Well here's one attempt: Divide the laboratory instructions into three sections (Prelaboratory, Activity, Postlaboratory) and call the computer into action.. Each student gets a lab. I.D. password and finds a terminal to the mainframe or a PC somewhere..

After a successful log-on and menu selection for the required experiment he/she sees confirmation that the session is valid and then:

> This is the prelaboratory section of Lab. X.
>
> You should already have worked the practice problems on this topic in your laboratory booklet and checked with your lecturer if you had any difficulty.
>
> You will now be given three questions each of which you must work through and input an answer to before the next is provided.
>
> They are chosen at random from a large list so your friends answers are almost certain to be useless..
> If you fail this exercise you will be allowed only one more attempt — but NOT WITHIN 30 minutes of the failed one. If you fail the second attempt you will be directed to see your lecturer.
> You will NOT be allowed to proceed with laboratory activity until you: (a) pass the prelab exercise OR (b) have permission from the lecturer.
>
> Press "enter" to display the first question..

The three (say) questions are each chosen to illustrate a different concept involved in the exercise. For each question there are several alternative problems illustrating the same concept and it is one of these problems that is selected at random.

Example Question 1. (not a typical Physics problem)

> How many dwarves did Snow White keep house to?
>
> (Enter your answer at the cursor position and press <enter> for your next question - no comments will be given on your answer. You may discuss your answers with your lecturer if you wish).

Provision is made here for confirmation that the intended answer has been correctly entered and an opportunity to reread the question is also given.

In this form of numerical answer, written responses are returned with a directive to input a numeric answer. If a calculation is involved from the data given the algorithm for this is used by the computer to check the student response.

Example Question 2.

> How many thieves do you associate with Ali Baba?
> a. None.
> b. One.
> c. Forty.
> d. One Hundred and one.
> e. Forty thousand.
>
> Choose the answer you think most correct, input its code letter at the cursor position and press "enter" for the next question - no comment will be ....etc.

In this multiple choice type question only upper case and lower case of the answer are accepted. All other forms Eg fully typed answers are redirected and the re-presentation of the problem is an option.

Example Question 3.

> What did Rumplestiltskin offer the captive Queen as a means of freedom?
>
> Type your complete answer in less than 10 words at the cursor position and press "enter" for your next etc.

The correct answer "Three guesses at his name" contains 3 key words that must be present. The computer searches for these and redirects numeric answers Spelling errors are allowed for Eg. gueses, gesses..etc and a list of synonyms for each keyword is provided.

After the last of the questions has been attempted the computer assesses the students performance and directs him/her to proceed with the lab. activity, try the prelab.. again later on or go to the lecturer for review of the prelab... questions. The usual pass level is set at about 70% and this seems to work quite satisfactorily. Successful and unsuccessful attempts are recorded as well as the date on which they were made.

Assessing the lab. activity is the most difficult part as numbers and units with tolerances and variations have to be programmed for. We have identified each piece of apparatus and ask the student to identify which piece he used via the code number on it. Results with this equipment are checked against results obtained by the lecturers using that equipment and acceptable ranges are provided.

It has proven beneficial to "modify" some pieces of apparatus so that the result of the laboratory activity is easier to handle. Eg Fo. experiments on the photoelectric effect the bandpass of the various filters used has been "fudged" so that a predictable straight line results. The gradient of this is tested against the "true" values and units are checked.

At present the program does not do a straight line fit to the students input points but just checks the gradient and intercepts he/ she inputs with the acceptable reference range.

Further calculations which have been made from gradients, intercepts and other parameters etc determined by the student are checked using the appropriate algorithm and data input by the student so that an initial mistake is not multiply penalized.

The post laboratory work is a single attempt system with usually 2 or 3 questions provided for the student to solve. Again these are chosen from a file of similar questions designed to test the understanding of the work that has been done in the lab.

The final output to the student is his computer assessment overall. A detailed assessment is provided for the lecturer showing all stages of the process..

Security is provided at a number of levels:

a. Student and account ID. This enables students to avoid others "practising" at their expense.

b. Code word for viewing of assessment lists - each viewing of the assessment list is also dated.

c. Code word for modification to assessment list outside a student session.

d. CPU connect times are logged against password and ID - this attempts to identify the "hackers" trying to crack the system...

The method has various degrees of applicability to different laboratory activities but the prelab and lab. sections can be made to apply to almost anything and of course there is no testing or feedback on report writing skills. However these disadvantages are more than offset by the huge saving in time that is achieved and the feedback on report writing is given on the laboratory experiments which are not computer graded.

Phill Higgins has been involved in Tertiary Education for almost 20 years and has lectured in most areas of Physics, some of Chemistry and Applied Mechanics and a few of Mathematics and Biology. His main teaching and research interests are in XRay analysis, Optics and Fluid Mechanics. He has been an active member on the Committee of the Gippsland Branch of the Australian Institute of Energy for several years and has recently become involved in Nurse Education from which his main incentive for CAL has developed. He is a lecturer in Applied Science at the GIAE

Athena, Andrew and Stanford — a look at implementation and evaluation in three large projects

Geoff Isaacs
Tertiary Education Institute
University of Queensland

Massachusetts Institute of Technology and Carnegie Mellon University have both embarked on enormous projects designed to implement workstation environments for use in academic endeavours. Project Athena at MIT is a network designed to service teaching and learning only, while the Andrew network at CMU is to service both teaching and research. Stanford University's computer-intensive environment for teaching and learning owes no allegiance to either the Athena or the Andrew model. Here we summarise the environments at these three universities and place them in context.

It is intended that this paper be the background to a presentation at CALITE 87, which will take place after the author has visited these three establishments in late September 1987. The emphasis of the presentation probably will be on the evaluation and staff development activities which are being carried out in these three universities in support of their CAL systems.

When one looks at the implementation of computer assisted learning in Australian universities - or even, by and large, in any one university - one generally finds a hodgepodge of efforts of varying generality, sophistication, successfulness and longevity. The methods and tools used are often totally idiosyncratic even between staff members within the same department and support from a central authority within a university is the exception, not the rule. A similar tale can be told of the evaluation of CAL materials and CAL-oriented courses.

There are overseas - especially in the United States of America - numerous large, well funded, centrally supported computer assisted learning projects within universities and it seemed to me that there might be some profit in looking at how some of them attack the problems of the implementation, support and, especially, the evaluation of CAL developments. I therefore decided to devote part of my Special Studies Program in the second half of 1987 to exploring these very issues in the USA. At the time of writing that exploration lies in the future; however I have now accumulated a considerable amount of information about the places I will be visiting and that information will be summarised here by way of background to the paper presenting orally at CALITE 87.

The universities on which I will concentrate are Massachusetts Institute of Technology in Boston, Carnegie Mellon University in Pittsburgh and Stanford University near San Francisco.

Why these universities? MIT and CMU are both hosts to multi-million dollar, substantially externally funded projects each involving the development of a campus-wide network system designed principally to support teaching and students. MIT is host to Project Athena and CMU is host to Project Andrew. Both projects are heavily subsidised by computer manufacturers who seem to see the universities as cheap resear :h and development labour and as possible test beds for new workstations and network developments. The projects are, to a limited extent, in competition, and both are currently making available to educational institutions either free or at a nominal charge software developed in their networking projects. Stanford, on the other hand, represents the non-aligned, but nonetheless computer-intensive universities. It is heavily involved in CAL developments and has a large central support system for CAL developments, but it has elected not to implement a network such as the Athena network or Andrew (yet?).

Carnegie Mellon University.

Even CMU's own documents speak of "Carnegie-Mellon's much ballyhooed effort to computerize itself as no institution has ever done" (Maguire,1986). The pattern at CMU has been to install a campus-wide computer network (called Andrew) linking many advanced computer workstations and high end personal computers into one coherent whole. The network and its workstations will distribute the load of educational computing, it is hoped, thus lessening the load on the university's large time- sharing mainframe computers and providing a more responsive computing environment for use in the education of students and for research and scholarly activity.

The CMU plan was first formulated in 1982 and involved the use of advanced function workstations having very high resolution graphics screens and the storage capacity and computing power to enable the graphics facilities (among other things) to be used effectively. These workstations, which might be made by any vendor able to meet the necessary specifications, were to be networked by a system having cabling to every room on campus. Eventually there was to be a workstation for every student (note, however, that contrary to widespread belief, students at CMU generally have not been compelled to purchase a computer), faculty member and member of the administration on campus - a total of some 7000 workstations. It was felt that the falling cost of workstations would make this plan feasible by the end of 1986, but in the event, the fall has not been as sharp or as great as expected.

IBM, in late 1982, got the project off the ground by contracting to spend $US 35 million to create CMU's Information Technology Center (ITC). ITC's purpose is to develop the software needed to make feasible CMU's plans for integrating computing into its educational activities.

It is hoped that the large concentration of computing resources becoming available to staff and students at CMU will stimulate the growth of a widely useful library of educational software. To help staff in writing such software (by funding, by advising or by actually getting it written) is the role of the university's Center for the Development of Educational Computing (CDEC).— Andrew

The name Andrew is sometimes used to describe all parts of the educational computing developments at CMU, but most frequently Andrew refers to the software base on which the developments at CMU are built; the best technical description of this system will be found in the paper by Morris and others (Morris et al, 1986). The Andrew software environment consists of three components: a high speed (4 to 10 Mbits per second) network, a file system to support a very large number (thousands) of workstations on that network and "a friendly, graphics- oriented operating framework for users of workstations" (Sherwood,1986) - that is, a user interface for graphics workstations.

The operating system on the computers which form the backbone of the network is called VICE ("Vast, Integrated Communications Environment"). VICE emulates a time-sharing file system; that is, there is one single file system for all stations on the network. It is integrated to such an extent that users may "access files in a uniform manner regardless of the specific workstations at which they are logged in, or where the files were created originally" (Morris et al, 1986). Changes to fiies are recorded centrally by VICE every few seconds regardless of where those files have been used. Devices like printers, bulletin boards and electronic mail systems are built on top of VICE and, like the file system, are available quite transparently to workstations users who need not know their physical or machine locations. Files in VICE being actually used by a user are in almost all cases copied to a cache on the workstation at which the user is currently working. Changes to these files are made on the copy in the cache and the file is written back to the common VICE storage area when it is closed.

The VICE file system is considered to be extremely secure because no user programs run on the computers which hold and control the VICE file system and file traffic carries only complete files from workstation to file server. In an ordinary network there would be an enormous amount of paging of file fragments between workstations and servers; however it was judged that the teaching and academic environments could generally be adequately serviced by a filing system model rather than the usual networked database model which allowed continuous access to the database for update and modification by several users. Where such databases are needed by Andrew users they are suppiied on an ad hoc basis by access to the university's pre-existing mainframe and mini computers.

Arms (1987) reports that in January 1987 there were 17 servers on VICE. Two of those servers were IBM PC-kT's and the remaining fifteen were Sun 2's and Sun 3's. Their total storage capacity was 17 Gigabytes. At the same time there were 3400 registered users with about 450 Unix workstations connected to the VICE system. This is only about half of the eventual target number of users.

Workstations under Andrew must satisfy a few conditions before being usable in the system. They must be fairly powerful (at least 1 mips), need a high resolution graphics screen (in the region of one million pixels), need substantial random access memory (in excess of one megabyte - preferably three or four), local mass storage (either a substantial fixed disk in the region of 40 to 100 megabytes or very fast access to a locally networked disk which emulates a local one), an Ethernet or a Token Ring interface; and they must be able to run the BSD 4.2 version of the Unix operating system (this last need automatically implies all of the others except the graphics screen and the network interface). Andrew as currently implemented does not allow the current generation of personal computers (IBM PC and AT, Apple Macintosh, Mac Plus and Mac SE) to be connected. A new version - Andrew Plus - will also allow limited use of conventional personal computers. Currently four types of Unix workstation are supported; they the Sun 2, Sun 3, IBM RT and Vaxstation II (see Arms, 1987). Andrew has been designed so that it is relatively easy to add support for other makes of workstation which satisfy the prerequisites set out above. Support for conventional personal computers, when completed, will allow them to treat the Andrew filing system as if it were a huge hard disk attached to the personal computer; an intermediate server (a Unix workstation) will sit between the personal computer and VICE and will hold the local copy of the VICE file being used by the personal computer. In early 1987 the IBM PC Server was complete and operational while the Apple Macintosh Server was expected to be available before the end of the year (see Arms, 1987).

The user interface environment running under 4.2 BSD Unix on the workstations is called VIRTUE ("Virtue Is Reached Through Unix and Emacs"). VIRTUE consists of a toolbox ("an object-oriented multi-media environment for applications development", to quote Arms, 1987) which handles the interface to the VICE file system and has a window manager that allows multiple processes to share the high resolution bit mapped display, and packages for manipulation of text and graphics. Currently Andrew uses its own, home grown window manager, however it is planned shortly to re-implement the Andrew user interface as layers on top of the more widely supported X Windows system developed at MIT. This is the first clear indication of what seems to be a convergence among some of the large university CAL projects, although communication between them has always been considerable.

Computer assisted learning at CMU

Naturally computer assisted learning long predates Andrew at CMU. The CMU catalogue of educational computing projects (Seiden, 1986) lists in excess of one hundred projects, some dating back to the 1970's, but most from 1984 onward. The target systems for these projects are as follows: IBM Personal Computer (some for PC, some for AT, various display adapters) - 39; Andrew - 33; Apple Macintosh (various models) - 4; other (mainly HP9836's running Unix in engineering departments) - 34; total - 110. Where a project was originally developed on another system, but has now been ported to Andrew it has been counted only under Andrew. These projects range from small BASIC language programs plotting graphs on a personal computer to large scale CAL authoring systems using all the facilities of Andrew, such as CMU Tutor. The bulk of the projects are in engineering, mathematics and science areas, but there is a significant number in the social sciences and humanities.

CMU Tutor

It is perfectly possible to use the facilities of Unix and Andrew to develop custom made computer assisted learning materials. The Andrew window manager has functions to allow user programs to handle text and graphics within windows and there are other, higher level programs to make this process simpler (the Grits database facilities, a graphics layout organizer and the like). However many potential clients of Andrew have neither the patience, nor the skills, nor the time to acquire them in order to use the C language and these tools to program their CAL applications under Andrew/Unix. CMU Tutor has been developed by Bruce Sherwood, Associate Director of CDEC and Judith Sherwood of the Mellon College of Science, together with other staff of CDEC and ITC in order to address these problems (Sherwood, 1985; Sherwood and Sherwood, 1985; Sherwood and Sherwood, 1986).

CMU Tutor is described by its implementors as "an integrated programming environment for advanced workstations", but it has now also been implemented on Apple Macintosh and IBM Personal Computers, although not all programs created on advanced workstations can be moved to these personal computers (the reverse does seem to be the case, however). It is an adaptation of the PLATO project's Micro-Tutor language (Sherwood and Sherwood, 1985) with extensions to support windows, mice, menus and other Andrew/workstation features. It is an incrementally compiled language (so lessons can be tested very soon after they have been written or revised) which supports all of the special text forms used by the Andrew editor in What-you-see-is-what-you-get format (bold, italics, font changes, centering etc) and includes extensive graphics capabilities including a graphics editor.

Massachusetts Institute of Technology

MIT's big thrust into campus-wide computer assisted learning started almost contemporaneously with Carnegie Mellon's. The MIT plan, announced in May 1983 was a five year plan "to explore new, innovative uses of computing in the MIT curriculum" (Lerman, 1987). MIT's plan was realized by the creation of Project Athena, a $US 70 million project financed by IBM and Digital Equipment Corporation to the extent of $US 50 million and MIT (by various fund raising initiatives) to the extent of $US 20 million. The support from DEC and IBM is support in kind (equipment, software, maintenance and staff support etc) not in cash.

Like the developers of CMU's Andrew, Athena's proponents would like software developed in the projects to be runnable on advanced function workstations from any vendor and have opted for Unix as the operating system of choice for those workstations.

Project Athena

What exactly are the goals of Athena? Lerman (1987) writes:

> Project Athena's mandate is to explore diverse uses of computing in support of education and to build the base of knowledge needed for a longer term strategic decision about how computers fit into the MIT curriculum. We encourage a wide range of educational applications to understand how computing helps students learn; we place computers in diverse settings to explore how different physical arrangements affect the use and value of computing; we experiment with different types of software to learn about its utility, reliability, the ease with which it can be run in a very large computing system and the costs of supporting it.

About half of MIT's $US 20 million contribution to Athena is being spent on the sponsorship of curriculum development projects. A project may be supported by direct funding, the supply of computational resources, the supply of programming assistance, the paid release of faculty from teaching or other duties. One hundred and eleven projects had been funded as at autumn 1986, almost exactly the same number as at CMU in a similar period. Funded projects need not be on a large scale - even small projects from individual staff members can be accepted.

Note that there is a major difference in philosophy between Athena and Andrew. The Andrew project is aimed at producing an efficient networked workstation environment which will facilitate the use of computers in education. It was felt that the provision of such an environment would provide the critical mass of support and resources needed to generate a significant growth in such usage. The Athena project is taking exactly the opposite approach. They are funding and encouraging an increase in the amount of educational computing activity in the hope that, at the end of the project's five year life they will be better able to design or select an appropriate environment for educational computing. Another major difference is that the Andrew project is aimed at the support of all persons on campus - students, academics, administrators - in the course of their work, while MIT's Athena is aimed mainly at the teaching of undergraduate students and, to a much lesser extent, at the teaching of postgraduates; it is absolutely not aimed at supporting staff in their research and scholarly activities.

Athena's catalogue of projects (Stewart, 1986) lists 60 in all. Of these projects twenty-five are described as being implemented on Athena Workstations (Vaxstation II or IBM PC-RT's mainly), eleven are implemented IBM PC-AT's, mostly in the "Athena configuration" (which includes an IBM Professional Graphics Adapter), six are on IBM PC's, and eighteen on other systems (generally, where it is clear what other system then it is one of MIT's large time sharing systems). Where a project has been implemented on both a PC and an Athena Workstation it has been classified only in the latter category.

This split of system types reflects accurately the development of Athena and its own philosophy. Since the aim was to develop a critical mass of computer assisted learning activity on campus, Athena's directors did not wait for the ideal workstation environment before sponsoring projects. Rather, they elected where possible, to move with an intermediate system - a heavily optioned IBM PC-AT - and, where even that was not possible, opted to use whatever usable system was available. Thus some of the projects in the 1986 catalogue are conversion projects, moving projects from Athena AT's or large time sharing machines onto Athena workstations. This generally seems to involve mainly conversion of the windowing and graphics facilities to Athena's standard.

Evaluation and Athena

There seems to be more evaluative activity of the educational aspects of Project Athena that there is at Andrew. This is, in part, a further consequence of the differing development plans for the two projects. Athena's emphasis on CAL software rather than network software means that there is likely to be more to be evaluated. Yet the difference is greater than that. Athena has actually created an Project Athena Study Group which looks at the effects of Athena on MIT. To date, however, the Group has published no papers or reports available to outside scholars. The Project also sponsors directly a longitudinal survey of undergraduates since 1985 in order to look at patterns of student use of computers. The most recent of these (Cohen, 1986) details the results of the 1985 and 1986 surveys. Cohen's report shows that a very high - and increasingly so - proportion of the undergraduate student body has used Athena (87 percent), such use amounting to over three hours per student per week on average. Interestingly, she found that usage by male and female students was equal in all classes and years at MIT, "...in contrast to male-dominated and senior dominated uses of other computers at M.I.T.". At present students' usage of Athena is much more likely to be personal (word processing and the like) than required for courses. Needless to say, most of the personal usage is course-related, however. Athena has entrenched itself in students' consciousness to such an extent that, according to Cohen, they "regard Athena as a service rather than as an experiment". This means, of course, that they criticise it as a service rather than as an experimental setup and this seems to be reflected in their complaints about crowded or inaccessible facilities, slow systems and system unreliability as well as low quality printers. It would seem that at this stage Athena's massive injection of funds and equipment has not solved the usual central computer service blues encountered by so many tertiary education institutions. To be fair, the Athena to which students are reacting at this stage is largely one of terminals to a time sharing system; it does not yet include the sophisticated workstations which presumably will lessen at least some of the problems reported above.

In addition to the large scale evaluations mentioned above there is also a significant amount of localised evaluation of materials and courses within individual projects sponsored by Athena. Some of this activity is mentioned in Stewart's (1986) compendium of projects, however few results of these evaluations have yet reached the open literature.

Stanford University

Stanford is not at the centre of a large externally funded project like those at CMU and MIT, but it is, nonetheless, a highly computer-intensive educational environment. However, possibly because it has no "big name" project, it is less well covered in the literature. All of the information presented here comes from an article in the February 1987 issue of Byte magazine (Osgood, 1987).

The centre of activity at Stanford seems to be IRIS -Instruction and Research Information Systems - and its attitude, as reported by its director, Michael Carter, seems more akin to the Athena philosophy than the Andrew philosophy:

What we are trying to do is enhance academic achievement by applying computer technology. Our best bet is to try to focus it a little here, nudge it a little there, lead a little bit over here. With so many really smart faculty members out there, I want to get them enough devices so that they know exactly what they want to do, and then follow them, rather than to control the way they use computers. The trick really is to remove the obstacles so that those people can lead the way.

Stanford has several "public" clusters of computers for student use (with students who are required to use them in their courses getting priority in access. Such clusters provide either personal computers only or a mix of personal computers and terminals connected to Stanford's time sharing system. There is also a number of classrooms at Stanford with PC's built in, some including large screens so that a whole class can view the same screen image. One gets the impression that at Stanford Apple Macintosh computers are preferred over IBM personal computers or compatibles.

IRIS supports CAL developments in several tangible ways. Among these is the Faculty Author Development program, under which IRIS has helped faculty to complete 36 software development projects ranging widely over the university. Such help may come in the form of varying combinations of equipment, design or programming support. Beyond such direct to aspiring CAL teachers, Carter's unit is trying to develop a range of self help tools to decrease dependence on IRIS for the preparation of CAL materials per se by making it easier for staff to write their own programs. The CAT (Courseware Authoring Tools) project, as this latter development is called, is being implemented on workstations similar to those being targeted by Athena and Andrew. Indeed, it is not the aim of IRIS necessarily to develop the required tools in house; they are quite prepared to borrow software from other projects, naturally including Andrew and Athena.

Conclusion

The real conclusions to this paper do not yet exist; they will be given in the oral presentation to which this written work is the background. It is hoped that the presentation will concern staff development and evaluation at the three sites named above, however, on the basis of our preliminary reading, we are not optimistic that there will be a great deal to report either from Stanford or from CMU. However appearances from the other side of the world can be very deceptive, so my forthcoming visit may produce some surprises in these areas.

All the above having been said, we can reach one significant conclusion on the basis of the earlier sections of this paper, and that is that there is a clear convergence among all three of these sites in the software and the hardware they are using. In hardware all three seem to be moving away from conventional personal computers and towards advanced networked workstations with very high resolution graphics and network connections. This reflects in part the increasing complexity of the tasks such workstations are being asked to carry out - in many cases supporting a multipurpose working and learning environment, not just running a few computer assisted drill and practice sessions - and in part the increasingly detailed graphics being required for tasks from those previously mentioned to supporting a "friendly" operating environment for the computer illit..ate. The great unknown in this choice of equipment is Apple's Macintosh II, which seems to be the only personal computer available in the marketplace at present (and it is only just available) capable of challenging the workstations on all counts; its future role at these sites may be clearer by the time I visit these sites in September 1987.

As far as software is concerned, the convergence seems partly the inevitable consequence of several sites independently pursuing the same ultimate aims with substantially the same set of hardware. But it is also the result of a freer sharing of expertise and of the actual software itself than might have been expected from the earliest publicity about Andrew and Athena. This in turn results from all of the projects being strictly non-proprietary in their target machines — Andrew would not be looking to use MIT's X Windows environment if it were targeted at one machine or owned by one manufacturer; it seems to have been adopted by Andrew not only because it is superior to Andrew's own home grown product, but also because it is increasingly being supported by equipment manufacturers. As an Australian I can only hope that these universities will be as generous with their CAL lesson software as they are at present with their systems software. Then my only problem will be to work out how to fund the purchase of a couple of hundred advanced function workstations - one senses that this is proving a greater problem than expected in the USA also.

References

Arms, W.Y. (1987) *Andrew-Plus. Carnegie-Mellon's Integrated Personal Computing Environment*, Carnegie Mellon University Academic Services, January.

Cohen, K.C. (1986) *Project Athena Student Survey Findings 1985 - 1986*, Project Athena Impact Study Report No. 5, Massachusetts Institute of Technology.

Lerman, S.R. (1987) *Questions and Answers About Project Athena*, Project Athena, MIT, February.

Maguire, A. (1986) Andrew is Alive and Well and Living in the Computer Network, *Carnegie-Mellon Magazine*, Spring, 15-19.

Morris, J. H., M. Satyanarayanan, M.H.Conner, J.H. Howard, D.S.H. Rosenthal and F.D. Smith (1986) Andrew: a Distributed Personal Computing Environment, *Communications of the ACM*, 29(3), 184-201.

Osgood, D. (1987) The Difference in Higher Education, *Byte*, (2), 165-178.

Seiden, P. (1986) *Carnegie Mellon University Educational Computing Projects*, 267 pages, Educational Software Library, Carnegie Mellon University, Pittsburgh, USA.

Sherwood, B.A. (1985) An Integrated Authoring Environment, *Proceedings of the IBM Academic Information Systems University AEP Conference*, Alexandria, Virginia, 29-35 (June).

Sherwood, B.A. (1986) Workstations at Carnegie Mellon. In *Proceedings of the Fall Joint Computer Conference*, Dallas, November 2-6, pp 15-17.

Sherwood, B.A. and J.N Sherwood (1985) *The Microtutor Language*. Stipes Publishing Company, Illinois.

Sherwood, B.A. and J.N. Sherwood (1986) CMU Tutor: An Integrated Programming Environment for Advanced-Function Workstations, *Proceedings of* *the IBM Academic Information Systems University AEP Conference,* IBM Academic Information Systems, San Diego, April.

Stewart, J.A. (1986) *Project Athena: Faculty/ Student Projects,* Project Athena, Massachusetts Institute of Technology, December 1986.

The computer based education system: A process of integration?

Michael McCrae
Department of Commerce,
ustralian National University

Current computer technology is fast becoming an important and widely used instructional tool. But managing the education process is much more than just instruction or assessment. This paper looks at the concept of an integrated computer based teaching system. At least three component areas of computer use may need to be integrated to produce comprehensive. student-centred learning systems -computer based instruction, computer managed learning and computer assisted course authoring. This paper offers an analysis of the advantages of integrating these areas to form a computer based education "system", rather than leaving them as isolated, unconnected 'application' areas. The limitations and requirements of such an integration process are examined and the likely feasibility of the concept opened for discussion.

Computer aided techniques now impact on a whole variety of tasks in the educational process. The penetration is becoming increasingly pervasive and sophisticated, both in design philosophy and task complexity. But the potential for the integration of computer assistance into a 'system' of integrated software products covering the total educational process under a computer based technological umbrella remains largely unrealised. On the surface the approach could combine integration with extension — generating and managing courses based on sound educational principles in diverse areas. Certainly the technical expertise and the necessa.y umbrella of technology is there. But attempts to produce complete, computer based education systems appear to have lagged behind technological capability. The proponents of integrated computer based education systems (CBES) argue their benefits strongly. But to date few comprehensive products have appeared, and only limited use made of the potential. Why is this?

This paper examines the potential of CBES as an example of an integrated approach to using computers in tertiary courses. Computers can also be used in the educational environment for various administration applications, but this paper concentrates only on educational applications.

The paper is organised as follows. Section two briefly examines the philosophy behind a 'systems' approach to computer based education. The section reviews some of the major functions and components typically attributed to these systems to illustrate the concepts of integration and extension (generality of application) which these systems offer. Section three examines the advantages and benefits claimed for an integrated approach to the educational process. Section four outlines limiting requirements and suggests possible shortcomings. Section five advances tentative conclusions on the future of the CBTS philosophy. The aim is to generate discussion of these issues based on participants individual experiences within tertiary education situations, rather than provide answers.

The CBES Philosophy

Current CBES systems grew out of extended computer based systems that are designed to manage assessment administration and recording keeping within the

the learning process (CML) (Rushby, 1985). Traditionally, most CML systems are based on a model of student study behaviour, and concentrate on managing reporting and assessment procedures in an attempt to allow the teacher to concentrate on teaching. They typically provide support in four areas:

- construct, mark and analyse tests for diagnostic/assessment purposes;
- keep student progress/performance records;
- provide individual guidance through structured course material;
- feedback reports on students performance and progress to teacher.

Using extended CML's as an interface, the CBES concept attempts to integrate course authoring, computer aided instruction, learning assessment and course management into an integrated suite of software packages. Typically three features characterise these systems:

1. The basic design approach views course design and implementation as an educational process which consists of sequential tasks that are purposive i.e. directed towards particular educational objectives. This contrasts with previous, segemented approaches directed at isolated tasks in the education process.
2. The design approach integrates the separate computer aided tasks into a system design philosophy. The emphasis is not just task compatibility but integration into a total system design.
3. The previous concept of computer managed learning is expanded to include the whole range of tasks which make up the education process in the design, construction and delivery of courses. Attention focuses on using the computer to actively manage the whole educational process rather than just managing CAI and assessment procedures.

The design concept of CBES is based on the principle that the process of course design, implementation and management is composed of a series of sequential tasks that can be defined and modeled as an educational process. The nature of these tasks and the use of computer and communications technology to assist their accomplishment is well documented (see, for instance, CBTS 1983; Denholm, 1984). Since common tasks are involved in all course design and implementation, computer based systems design can be used to manage the learning environment, and to provide the integrated environment for developing CAI programs where these are evaluated as being desirable and necessary. The integration of CAI, CAS and CML functions under one technological umbrella has been designated CBTS or CBES. The emphasis is now on integrating separate tasks as part of a total system.

Historical Perspective

The historical impact of computer technology on the tertiary education process is one of ever wider and more sophisticated applications. The progressive steps in the progression include[1]:

- use of computers as an instructional aid as an outgrowth of programmed learning (Rushby, 1985);
- growth of interactive instructional dialogue;
- using the computer to aid or control design of the instructional process, including course authoring;
- using computers to manage the learning process;
- the development of integrated CBES.

These steps reflect the development of computer aided instruction (CAI), lesson and course authoring systems (CAS), computer managed course implementation (CMI) and then using computers to manage the whole learning process (CML). Of course the definition of the tasks performed by these categories will vary between place and author; especially for instance in the scope of computer assisted instruction (CAI) [See, for instance, Robbins (1983)].

McDevitt and Price (1984)provide an alternative view of this progression. They suggest that the development path taken by computer based teaching systems reflects the general evolutionary pattern of information processing technology. This progresses from the Transaction Processing Phase, through the Information Processing Phase, to the Data Management Systems phases.

This proliferation of computer aided task performance has several features relevant to the present discussion:

- Evolution of software for computer based course authoring and managing has been superimposed on the more traditional 'passive' role of computer aided instruction;
- A progressive process of bringing together technological advances in several areas including information technology, communications, and networking;
- Improvements in understanding of the analyticals and design of course presentation, design, learning and the educational environment;
- Increasing analysis of educational objectives and their effective attainment through computer assistance;
- Analysis of the education process into separable tasks, and breakdown of the components of those tasks;
- The development of increasingly complex sets or suites of software that were compatible with one another;
- Often built by those in charge of courses, rather than those skilled in the theory of education and learning;
- Much reinventing of the wheel due to separate developments of functionally similar software suites in different institutions, and even within institutions;
- Specificity and lack of transferability of applications software. Applications often limited to specific courses or computer systems.

The developing sophistication of many of the computer based course authoring aids also focused attention on the need for computer involvement in the 'active' areas of course design and management followed by management of the learning process rather than just 'passive' computer use in already designed lessons or courses. Many of the essential ideas of integration, extension, system and educational process are present in these developments (Harris, 1983). For instance Sutanto(1984) gets near to the concept in his Computer–Based Teaching System in Power System Analysis.

The motivations for movement from a 'passive' role of computer usage for computer aided instruction towards a more 'dynamic' involvement in managing the teaching process may be complex. But economics is never far away. Tertiary education is notoriously short on funding. The expensive nature of CAI soon emphasised the need for strategies, since CAI can be a costly tool to devise. The following critical questions began to be asked:

- what should be CAI'd
- which students should get CAI
- reducing cost of locally developed CAI
- structuring CAI lessons to maximise local applications.

All of this implies the need for development of CAI strategies aimed at cost effective results and maximisation of educational impact. An "engine" is required that will initiate and manage the C. process for maximum effectiveness within budget constraints. Denholm (1984, p. 38) points out that "t implement an individualised learning environment with present day budgeting constraints or when just effectiveness of effort is sought, CAI is probably best applied via a Computer Managed Learning Program."

But even prior to the development of an 'integrated' approach to computer aided education, Earle (1984) notes that in the early '80's Leiburn (1981, p. 277) summarised the general benefits of Computer Based Teaching as : "individualisation, self-pace learning, immediate response and knowledge of results, automatic collection of performance records/evaluative data, patient, tireless tutor or drill master, active concentrated learning, motivates and reinforces learning, cuts learning time, gives more time to instructor for providing individual assistance, relieves instructor of rote/repetitive tasks." Given that these benefits were well recognised, what does the philosophy or 'system' design concepts that underlie CBTE offer that is new or innovative? What differentiates it from previous developments in computer aided education?

'Integrated' Functions performance

The essential purpose of CBES is that computer performance of teaching and learning tasks should be integrated into a computer based system of education. In practical terms this means that separate pieces of software written to help perform various tasks in the educational environment could be integrated together into a software 'system'. The design of each task would then take place within an overall 'system' design for a computer learning environment. The system can then provide a set of tools that articulate to provide a system with the widest applicability to diverse course and educational processes in tertiary environments. The system models the education process as a schematic whole, rather than concentrating on computerising separate tasks. It is the system philosophy as an approach to incorporating computer aids within the educational process which is important[2].

The approach integrates the tasks of:

- designing courses and authoring course content
- Instructing the learner (using computer aided teaching)
- managing the learning process.

Looked at in this way, the CML interface can help give the system a generality in application to diverse courses, since there are common tasks and task sequencing within this subsystem that may apply to all applications. An extended Computer Managed Learning System (CML) becomes an integrating interface for other subsystems in the total process. It represents an applications software product designed to manage learning activities in an educational environment. It provides an integrated orchestration of the learning environment. The CML in a functional sense becomes the environment in which learning activities are managed, pointed towards objectives, and in which CAI takes place. The functions it supports include:

course authoring
course organisation
student testing
instructor management facilities
evaluation of program effectiveness
management of student records
integration with CAI
communication with microcomputers
development of courseware data base

An integrated system can have many contributing applications. It is the possibility of of joining components together which makes them dynamic and fully of potential. Try (1984) includes the following application areas:

- Curriculum planning
- Curriculum management

- Diagnostic/Prescriptive teaching functions and Assessment and reporting
- Catering for individual differences
- Using the computer as a learning resource
- Administration

CBES can provide computer support to educators and trainers in the development and routine maintenance of multi-resource, multi-learning environments. To achieve this system design includes the inherent characteristics of:

- program independence
- content flexibility
- value independence
- terminal efficiency
- real world learning
- courseware data base independence
- multi-level security

Importantly, CBES provides the framework within which instructional development, delivery and learning evaluation processes occur using computer based methods.

Advantages and benefits

The benefits of integrated computer based education systems are well documented in the literature (See for instance Rushby 1985; Chesmond 1984; Digital 1984; Try 1984; Sutanto 1984). These benefits may be classified in terms of:

- the learner
- the teacher
- the organisation

Learner benefits include:

- curriculum targeted to his needs
- learning programs independent of other learners
- accurate and timely control of the learning experience
- progressive insurance of mastery over steps
- efficient learning resources
- technology training
- a one-to-one interaction of student with computer
- students can proceed at own pace
- presents problems in logical order
- training can be individualised

These advantages support the learner by:

- providing for individualised learning programs
- directing and tracking learner progress
- generation and marking of tests
- providing feedback of results and remedial teaching

Benefits for the teacher include:

- maintaining records
- providing course status information
- providing a framework for course development and organisation
- providing a communication link with learners
- managing progress evaluation and reporting.
- course material can be updated conveniently
- enhances rather than substitute for lectures
- records student performance for assessment.
- provides practice and remedial help until the desired level of mastery is reached.
- student's record and criticism can provide information for improving future changes in course material

These advantages combine to:

- time saved through shifting of clerical and routine tasks to the computer. — e.g. report preparation, record keeping, statistical analysis and marking.
- an integrated system means the teacher can focus on using the computer applications and not be concerned about technical aspects of equipment or software management.
- Overall, more time can be spent planning, developing and teaching and less on administrative and technical tasks.

The benefits to the tertiary organisation include:

- finely tuned curriculum or course planning
- flexible and adaptable course planning facilities measurement of attainment of course outcomes
- powerful individualised learning systems

The desired objective arising from these advantages of using CBES is increased educational products and better achievement of educational objectives per educational dollar spent.

Educational Advantages.

This list of benefits is a summary of those found in the literature. They are well known and add nothing new. But several of the potential educational benefits have not been explored in detail, and since they do arise from the uniqueness of the CBES approach are worth discussing and evaluating. Two educational advantages illustrate this point.

1. Course design and implementation is seen as a complete learning process composed of educational objectives and particular elements that should combine into an effective learning environment that is purposive; i.e. whose main objective is to achieve effective and efficient learning by the student within the cost and resource constraints operating in that particular environment. There is of course, nothing inherently ineffective in mobilising computer based techniques into specific areas or tasks within this overall process. It happens all the time in computer based course authoring and course management techniques. But this 'component' approach may not encourage a 'system' view of the educational process. A view that sees course delivery as a rational, purposive integrated multi-task process aimed at effective learning. A process where each task or function in the educational process meshes to produce an educational process greater than the sum of its parts.

2. It emphasises the need to inject learning and education theory into course design and implementation and evaluation. Many tertiary educators, while experts in their field are not educational experts well versed in the principles of effective education. This can be a perennial problem in universities, especially in Australia, where techniques of formal evaluation of effective teaching are not in vogue, as is the case in some other countries such as USA. Thus, tertiary educators expert in computer based technology, or well versed in its application to course situations may use computer facilities to aid the educational process in various areas without attention to principles of sound education. The danger is in perpetuating ineffective educational processes.

This problem of coincidence of educational theory with computer technology expertise is a continuing one where computer aided learning processes are devised by individual tertiary educators in their particular fields. Use of computer power in particular courses may give the appearance of more learning for each education dollar spent, but this may be at the expense of the continuation of ineffective teaching practices. I am not arguing that this is always the case. On balance the injection of computer based learning or management techniques into course structures and implementation might be expected to consistently lead to improved effectiveness and efficiency in terms of educational objectives satisfaction if for no other reason than that the change involves an in depth look at what is being taught, how it is being taught and the learning evaluation processes. The point is that this review and improvement process may be optimised where a 'system' view is taken of the educational process which then attempts to base the total system design on generally accepted sound educational principles.

Effective CBES systems design would seem to necessarily require the incorporation of sound educational principles. So that, to the extent educators use such systems in course preparation and management, they follow paradigms that take account of accepted educational principles for improving or optimising the learning process, even though they themselves may not possess such skills or be experts in educational processes. If the system is based on such principles then the degree to which the course convener and presenter is trained in theory of educational process design becomes less critical to the design effectiveness of his courses. A CBES may help to separate or disengage educational design skills from skills in the particular discipline area being taught. The former is embedded in the CBES, the latter remains the province of the course convener.

This advantage of course, is dependent upon the extent to which particular CBES incorporate such principles and points to the importance of the input of experts in education and learning theory into the CBTS design process. If CBTS is designed solely by information system and software technocrats who merely mechanise existing course and educational structures, then to this extent, these benefits may not be realised, and the system may merely perpetuate ineffective teaching, learning and evaluation procedures.

Limitations

The typical limitations of CBES are echoed in Sutanto's (1984) reference to the limitations and disadvantages of the PLATO, and in an earlier but similar similar evaluation by Montanelli (1979). They stress:

Resource Costs

- Resource requirements may be substantial, even prohibitive to Departments with little computer system capacity;
- Programming requirements are onerous, especially graphics component; • Frequent use by substantial student numbers is necessary for adequate return on investment;
- Expense — expensive to lease 'PLATO' terminal;
- Extensive capital investment to obtain whole PLATO system.

Problems of standardisation

- Curriculum differences in specific courses between universities require courseware changes
- Only runs on a Cyber computer system
- Software source code not available.

Sutanto's conclusion highlights the compromise between resource limitations and effective education. CBES can dramatically improve teaching however "one cannot justify the extensive cost in obtaining PLATO system purely on a pedagogical ground especially with the shrinking budget of most Australian universities.

Conclusions

In the enthusiasm for computer aided tools, it is easy to forget that the computer is only one of many tools in the education process. The learning environment and experience could be deprived of much of its richness if the diversity of teaching/learning instruments and experience were replaced solely by computer based education. Such criticisms of computer dominated educational processes are valid. However, they are essentially based on a 'conflict' perspective of computers in education. The idea that computers inevitably replace or supersede other instruments; even including the teacher. The shadows of 1988 are still not far away.

Taking an opposite view, it is possible to argue that used with sensitivity and appropriateness CBES may not 'conflict' with, but enhance the use of alternate teaching techniques and instruments. This supportive role can be particularly argued in relation to the use of computers as an integrating tool within course design, construction and implementation. For example, it offers possibilities for:

- Introducing sound learning and educational principles into basic design and approach to tertiary courses by emoedding them within course CBES design;
- Relieving the course convener and teacher of routine, recording and administration tasks;
- improving the efficiency in course design by giving the convener and teacher a basic paradigm to refer to in the design process;
- An underlying planning system for the teacher which integrates all the tasks in course construction and presentation, while leaving the teacher free to use those educational instruments for presentation, learning and review which he sees as appropriate.

Unfortunately, such benefits are often difficult to quantify in money terms; and even more difficult in terms of money benefits per period. Cost is superficially more easily expressed in dollar terms. Consequent, especially in the present climate of depleted funding for the tertiary sector, it is the absolute cost of educational enhancements which determine priority and commitment. This applies to CBES. Unfortunately, present systems involve substantial initial outlay either to buy or to participate in a bureau type service through remote access. If CBES purchase or access could be made inexpensive relative to the levels of present day Faculty or Departmental budgets this may encourage wider development or participation. I suspect that the reason most CAS, CAI and CML developments are by interested individual course conveners and Faculty members is the lack of funds to finance such initiatives out of Departmental teaching resources.

Perhaps a partial solution to this common dilemma may be the introduction of less expensive cut-down versions of CBES systems for micro computers. Such versions, while cheaper, do not necessarily have to sacrifice the power of the system in terms of tasks performed. The limitations are more likely to come in limitations on memory capacities, so they do not suit courses which would have extensive memory requirements. The key here may be in devising high productivity systems for microcomputer applications. Systems that are inexpensive and compatible with the type of memory capacity, data base capability and processing power now becoming more widespread among Departments as they take advantage of microcomputer systems.

Footnotes

1. Of course these developments may and have continued in parallel as well as sequentially. For instance, in the area of computer aided instruction, developments have occurred in at least four CAL paradigms — the instructional paradigm, the revelatory paradigm, the conjectural paradigm and the emancipatory paradigm. For details see Rushby (1984).

2. This approach moves one step closer to a concept of a generalised expert system of the educational process for tertiary courses.

References

CBTS (Australia) Ltd (1983) *A Comprehen-*

sive Computer Based Education System, Melbourne: CBTS, Documentation.

Chesmond, C. (1984) A Role for the Education Sector in Developing Computer-Based Training Packages for Industry, *Proceedings, Calite,* Brisbane, pp. 8–20.

Digital, (1984) *Introduction to Computer-Based Education,* Digital Equipment Corporation.

Denholm, R. (1984) Applied Economics to Make CAI More Viable, *Proceedings, Calite,* Brisbane, September.

Earle, T (1984), Development and Implementation of CML in a Computing Course, *Proceedings, CALITE,* Brisbane.

Harris, M. (1983) What an Authoring System SHOULD be Like, *Proceedings, Calite,* Brisbane, pp. 432–445.

Jones, A. and O'Shea, T. (1982) Barriers to the Use of Computer Assisted Learning, *British Journal of Educational Technology,* 13(3), 207–217.

Leibum, M, (1981) Factors Sometimes Overlooked and Underestimated in the Selection and Success of CAL as an Instructional Medium. In R. Lewis and Tagg (eds) *Computers in Education,* North Holland, p. 277–283.

McDevitt, A. and Price, J. (1984) dBase II: A Tool for Computer Aided Learning, *Proceedings, Calite,* Brisbane, pp. 167–187.

Montanelli, R (1979) Evaluating PLATO in the Teaching of Computer Science, *Journal of Computer-Based Instruction,* 5(3), 72–76.

Robbins, G. (1983) Tomorrow to Fresh Woods and Padtures New, *Proceedings, Calite* Conference, Brisbane, p. 327–9.

Rushby, N. (1985) *An Introduction to Educational Computing,* London: Croom Helm.

Sutanto, D. (1983) Assessment of the Use of 'PLATO' in Electric Power Engineering Education, *Proceedings, Calite,* Brisbane, pp. 371–5.

Try, K. (1984) Computers in Education: An Integrated Approach, *Published Proceedings, Second Australian Computer Education Conference,* Sydney, September, pp. 214–5.

Education resource planning

G. C. O'Brien
School of Economics
La Trobe University

In recent years there has been an increased need to match occupational training at all levels with the perceived or forecast labour requirements in the associated occupations. This has been especially so in the TAFE area. This paper explores the theoretical aspects of matching occupational requirements to enrolments in the associated educational programs. It then describes an implementation of this theory in an interactive computer package which matches occupational requirements in Victoria with TAFE courses which lead to occupational qualifications. The package has been developed for the Victorian TAFE Board and forecasts the increase or decrease in enrolments required to meet labour requirements in each occupation at the end of the century under various economic scenarios. These ideas can be implemented at all levels and would be a useful tool in University and CAE Careers Units as well as for Education planners. This paper emphasizes the design philosophy and other aspects of particular interest to Ascilite members

In recent years there has been an increased need in all sectors of education to match enrolments in specific educational programs with the needs of our society. In particular there has been an attempt to match the resources put into education to the requirements for specialists in particular areas of the labour market some years later when the increases or decreases in enrolments affect the size of work force. It is often difficult for planners to understand the implications of lags in a dynamic system. Indeed it may prove necessary to increase enrolments in a particular educational program to meet a target level of employment in some associated occupation at some future time even if that future target is lower than the current employment level. This could happen for example if current enrolments are not sufficient to meet replacement requirements.

Economists are concerned with matching supply and demand, but there have been few attempts to do this in educational planning because of the difficulty of matching specific occupations to specific educational programs.

This paper describes such a match and discusses both the theoretical considerations and the practical difficulties associated with such an exercise. These ideas are then incorporated into a practical microcomputer implementation which is designed to work on an IBM style personal computer and could be easily implemented on any other machine. The program provides a host of facilities which would be useful not only for planning purposes but also as a resource for career counselling units.

Program Planning Requirements in Technical and Vocational Planning

The system described in this paper, while generalizable, relates specifically to the provision of technical and further education in Victoria and to the associated occupational groupings in Victoria.

The Victorian TAFE system, in common with the rest of Australia, offers a wide range of courses through schools, colleges and adult education authorities. These courses may be classified into six streams: professional, para-professional, trades, other skilled, preparatory and adult or further education. In 1985 there were 330,000 persons enrolled in these six streams, the largest stream being adult education with 131,000 enrolees, while 31,200 were enrolled in the basic trade stream. The trade stream is exclusively apprenticeship in nature and accounts for nearly one-quarter of total contact hours. This stream leads to trade qualifications and registration and is the subject of this study.

The basic trade stream comprises some 117 separate courses which lead to 52 proclaimed trades. Individual course enrolments can vary from a handful of students studying at a particular location to courses such as motor mechanics which can have thousands of students enrolled at dozens of different locations.

The size and complexity of the TAFE system in Victoria - an annual recurrent budget of over $200 million, 17,500 staff (6,300 full time), 32 colleges and 600 smaller providers - creates significant co- ordination, management and planning problems. Apart from the size and complexity of the system, the high degree of autonomy of the colleges, the main providers in the trade st[illegible] obviously exacerbates these problems.

Since 1981 when considerable autonomy was granted to the separate colleges, TAFE has had to revise its planning processes. Part of this process is the development of a management information system to provide the necessary data for its internal needs. The system discussed here is a key element in this system.

Educational and Occupational Trends

As about 70% of TAFE's recurrent budget is devoted to its vocational activities, it is obvious that a more effective and efficient system will necessarily take into account labour market data in planning new programs or in expanding or contracting existing programs. The trades group of courses to a large extent require specialist facilities and staff who can not easily be reassigned to other programs. Hence significant changes in the mix of occupational requirements in the community will have significant implications for these 'facilities bound' training programs.

It was realised that projections beyond the year 2000 would not prove very useful in practice, hence the design philosophy was to develop a system which examined a 15 year 'planning period' beyond the base year. On the understanding that there would be continued revisions of future occupational requirements projections for trade and technical groups a base set of projections were required. These were found in a study of industry and occupation work force forecasts for Victoria under various economic scenarios for the 1990's.

In this analysis four different macro-economic growth scenario are available. These are

1. parity with OECD growth
2. parity with world growth
3. parity with high world growth
4. parity with low world growth.

These different growth scenario together encompass the range fr[illegible] pessimistic t[illegible] optimistic estimates and hence in the final model provide a sensitivity analysis for various kinds of growth.

It must be stressed that estimates of occupational requirements are not measures of occupational demand as they do not include any replacement demand. These occupational requirements indicate only what work force skill composition will be needed if the growth and output levels are to be attained. To estimate occupational demand the age composition of the work force was used to generate a Markov replacement process, and details of this are given below.

Units of Analysis

The fundamental difficulty with linking occupational requirements with training programs which lead to these occupations is that there is not a one-to-one relationship between these two sets. Under any of the classifications of occupations there are some which have no associated instructional program in the TAFE system, some have one associated instructional program and others have more than one instructional program. On the other hand there are some TAFE programs which lead to more than one occupation. The concept of a 'unit of analysis' has been developed to allow a rational linking or 'crosswalking' of occupational groups and instructional programs.

A unit of analysis is a mapping which associates groupings of educational/training programs, having common curricula and objectives, with the related occupations for which they prepare program participants. The development of this mapping or set of units of analysis is central to any attempt to link occupational supply and demand data. The absence of such an interface means that the separate elements of data collected on the supply and demand sides of the labour market are of limited use for program planning. Indeed, such a link is a necessary pre-requisite for any consideration of supply-demand balances for regions and occupations. Of course there is as much art as science in determining this mapping and it is sometimes necessary, on the grounds of either utility or empirical validity to discard some occupations, and, more rarely, instructional programs from either the range or the domain of the mapping. These issues are pursued in the report to the TAFE Board.

Units of analysis are formally classified as follows:

Type I — one program and one related occupation

Type II — one program and several related occupations

Type III — several programs and one related occupation

Type IV — several programs and several occupations.

A complete supply- demand interface will clearly contain, in general, all of the above types of units of analysis. Previous attempts at similar analysis have been restricted either to a single education- training program based supply-demand interface (Types I and II only) or to or a single occupation based supply-demand interface (Types I and III only). Both of these formats force the user to impose breaks by excluding certain occupations from particular programs. Often the excluded occupation-program groups are those streams with the largest enrolments. Our approach makes no exclusions on the grounds of analytical expediency, although we have made some minor exclusions where no data or surrogate data is available.

Career Progression and Replacement

Unlike some other countries there is a dearth of data in Australia on student progression rates, post-program labour force entry experience, occupational mobility and labour force withdrawal for trades based training programs and occupations. Some studies have attempted to describe tradesperson's career paths but unfortunately aggregate over trade groups and provide significantly different progression rate estimates. There is a significant gap in our knowledge of the immediate post-trade training entry experience of individuals. A further major gap lies in our knowledge of progression rates by year or level through the apprenticeship program itself.

In this study the estimates of year-to-year progression rates are derived from enrolments over the last few years. TAFE has only collected comprehensive enrolment statistics on a regular basis since 1981 and this period has also seen a rapid growth in the variety and size of trade related programs. At this stage it is not possible to track students over time so our year-to-year or level-to-level survivorship rates by trade stream category are necessarily tentative. TAFE is currently supporting pilot longitudinal tracking studies, but the partial information which does exist indicates not only considerable variation between courses but also that these progression rates are sensitive to external labour market conditions and vary considerably from year to year. Our system enables the user to modify these progression rates for each trade course, and of course this allows an experimental determination of the sensitivity of the predicted required enrolments to variations in these progression rates.

The entry rates to the trades following training are even less well understood. It is clear from Sgro's study that demand is sensitive to wage levels but our estimates of this entry rate, while plausible, have little empirical evidence to support them. Once again these rates are able to be modified by the user and experimentation with different rates is possible.

There is also a limited amount of information available on the career paths of tradespersons. It is very difficult to obtain any estimates of such key variables as: the proportion of trades qualified persons who have never practiced their trade, the age specific proportions of practicing tradespersons who stop using their trades skills and the age specific rates of permanent work-force separation. Certainly this information is not available at the level of individual trades but a recent ABS survey provides some base line data which enables us to make some preliminary estimates.

This survey by the Australian Bureau of Statistics is an Australia wide multi-stage area sample of private and non-private dwellings. The sample size and the nature of the survey limit its usefulness at the individual trade level, but it does provide a valuable framework. Of the more than one million trades qualified persons in Australia only 56 percent were employed or working in their trade or were using some trade skills in their present occupation. Other key findings, from the point of view of our analysis, were that :

this figure of 56 percent varied markedly from trade to trade (from 8 percent in footwear, clothing and textiles to 71percent in the electrical trades),

an estimated 5 percent had never worked in their trade,

the proportion still practising their trade declined sharply with age,

by age 25 approximately one quarter have left the trade.

Other studies cast little light on these statistics, some providing similar estimates and others contradicting this study. All of these other studies were smaller in scope and size and provide only partial estimates. Once again the facility has been provided in the computer program to undertake experiments by varying the consequent cohort-to-cohort progression rates.

In the ABS career path survey age specific retention rates are given for age cohorts, most of which span 10 years. These published rates for each trade have been subjected to a sixth order filter with the appropriate boundary conditions in our study.

This process provided estimates of retention rates in the first few years in the trade and subsequent analysis matched the results obtained using ten year cohorts. This approach has the disadvantage of being relatively arbitrary but appears to give robust results in conformity with the published data. There is, however no way of extracting from this data transition probabilities from the training program to the trade. The question of later entry to the trade is, of course, no problem since we are only concerned with modelling effective retention rates. Sensitivity analysis, either experimental or theoretical, provides some measure of how critical these potential inaccuracies are to the final estimates. A series of cross-sectional analyses at different times could also provide some estimates but to our knowledge no such study has been done.

Modelling Throughput

The methodology behind our analysis is general systems theory, a branch of mathematical analysis which includes the theory of Markov processes. Initially we assume that the parameters which describe the system are invariant over time, but this is relaxed in the most general implementation of the analysis. In practice it is a relatively straightforward matter to determine whether this restriction to an autonomous system is critical.

Systems theory relates the output of any system, in this case the number of qualified tradespersons in any future year, to the inputs to the system, in this case the number of entrants to the appropriate TAFE program or programs. The state of the system is the number of tradespersons or apprentices of each age in each trade group, which is denoted by where the subscript i denotes the industry, the subscript j denotes the level in the instructional program for j = 1,...,4 and the number of years in the industry for j = 5,...,15 (j = 5 corresponding to the first cohort, j = 6 to the second etc.,) and k denotes time. In the present formulation k = 1 corresponds to 1987, but this is easily changed.

The dynamics of the process are determined for each industry by the matrix of transition probabilities whose jl -th entry is the survival probability from state j to state l at time k, and the inputs to the system by the vector whose entries correspond to new entrants to the system into the levels of the TAFE course or into age the cohorts of the trade occupation.

The process is then completely specified by the equation which enables us to predict the number of tradespersons or apprentices at any given level or in any age cohort, employed in industry is at anytime from the present:

Making the calculations determined by the preceding equation is considerably simplified because of the special nature of the Markov transition matrices and , as mentioned above, we initially make the additional simplifying assumption that the transition matrices are invariant over time. However the advantage of the above general formulation is that entrants to the industry other than first year TAFE trainees can be incorporated. There are some further technical difficulties associated with matching single year cohorts in the TAFE program with a mixture of five and ten year cohorts in the trade but those details will not be discussed here.

From this general formulation it is not difficult to determine retention rates and age specific profiles for each of the trades and more importantly, in terms of later economic analysis, the average working life in the industry for each of the trades in question. These calculations also provide a verification of the estimated transition probabilities in each trade by the following consideration. The system can alternatively be modeled on the basis of retention rates and the transition probabilities in equilibrium determined as the Frobenius-Perron maximum eigenvalues of the dynamic system determined by the retention matrix. This alternative formulation provides an heuristic verification of both the analysis and the data.

This analysis is then used to provide estimates of required enrolments in the TAFE instructional programs associated with each unit of analysis to achieve specified target numbers of tradespersons in the associated occupational group at some specified future time. There are obviously many different adjustment processes but only two are implemented in our formulation of the model. These are

a. a constant annual increase or decrease, as appropriate, in first year enrolments, that is, where E is the constant annual increase in first year enrolments, and

b. a constant annual rate of increase or decrease, as appropriate, in first year enrolments, that is, where r is the constant annual rate of growth of first year enrolments.

Both of these growth scenario are useful to the planner. The first requires a constant annual increase in accommodation, equipment, staff and resources while the second requires a constant annual percentage increase in accommodation, equipment, staff and resources (when growth rather than decline is predicated). Both approaches have their adherents in planning. However in the case of a substantial adjustment to the enrolment numbers being necessary it is important to understand that the second scenario will, in general, create the largest overshoot. This is particularly the case for those units of analysis with a four year apprenticeship. In this case, increased enrolments do not begin to affect the stock of tradespersons until four years after the commencement of the program, and enrolments in the last four years do not affect target occupational levels. It seems sensible therefore to reduce enrolments in the last four years to replacement levels. However in some cases this would produce drastic enrolment changes which could not be accommodated by the system.

Structure of the Program

The implementation of the above theory and associated practical considerations has resulted in a Turbo Pascal program which serves an eightfold purpose. The program has been designed to be used interactively and provides a fast, user-friendly operation. It is completely menu driven and requires no user manual, at least for the target audience of TAFE officers.

The first objective was to provide an easy introduction to the concept of units of analysis and to allow these to be accessed firstly from one of the associated occupations, either from the older Australian Classification of Occupations (NCO) or from the new skill- based Australian Standard Classification of Occupations (ASCO), secondly from any of the associated TAFE instructional programs, and thirdly directly from the unit of analysis itself. The program displays the complete identification associated with the unit of analysis, detailing the official codes and titles for each element of the map. Help screens are built into the system to provide sufficient information for the first-time user.

The second objective was to provide a complete easily accessible catalogue of all of the ASCO and NCO trade classifications and all of the trade related TAFE training programs. This catalogue allows the user to browse through these lists for whatever purpose or to select an associated mapping or congruence from any entry in the catalogue.

The third objective was to provide on-line access to a statistical data base relating to the total number of tradespersons in any trade group, the age distribution of these tradespersons, the career transition probabilities and survival rates across age cohorts, the enrolments at each level of each TAFE trade course in the base year, the transition probabilities from level to level and from the final year to the trade.

The fourth objective was to forecast target occupation levels for each occupation group in 1995 and 2000 on the basis of four different economic scenarios for Australia, and to relate these targets to the units of analysis.

The fifth objective was to analyze the given data to determine the required enrolment levels over some planning period to meet a selected target number of tradespersons employed in the selected trade, either in a scenario of constant growth or of a constant growth rate.

The sixth objective was to determine the replacement requirements for a particular target occupation under a chosen growth scenario, the average age and the age distribution of tradespersons at the terminal time. It is important to emphasize that since this analysis is to a large extent concerned with disequilibrium adjustment processes then replacement requirements, the average age and the age distribution of tradespersons will vary from year to year.

The seventh objective was to allow all of the parameters in the model to be modified by the user either for the purpose of conducting experiments or sensitivity analysis or of modifying the parameters as calculated from other studies or available data.

The eighth objective was to enable the addition of further occupations and training programs and the consequent definition of additional units of analysis, if it proved necessary.

Conclusion

The program as demonstrated at Calite-87 fulfils the first seven of the objectives specified above although the author has deemed it expedient not to provide access to the parameter modification module at this stage. The nature of non-autonomous dynamics is not easy to understand and some skills are required to interpret data from the simulation of such systems. The program has been designed to allow the easy incorporation of additional occupations, instructional programs and further data but some modification to the source code will be necessary.

Acknowledgement

The author would like to thank David Wilkinson from La Trobe University for testing the program and identifying several bugs.

Bibliography

ABS Report, (1984). *Career Paths of Persons with Trade Qualifications Australia, September to November, 1982,* Canberra: Australian Bureau of Statistics.

Frugoli, P.A., (1982). *Guide to Forming Units of Analysis*, Washington: National Occupational Information Coordinating Committee.

Langley, P.C. (1985). *Employment and Occupational Trends in Victoria, 1985- 2000,* Adelaide: TAFE National Centre for Research and Development.

Langley, P.C. and O'Brien, C.C., (1986). Modelling Trainee Throughputs, Western Economic Association Annual Meeting, San Francisco, July.

Sgro, P.M. (1985). Apprentice Demand and Workers Compensation Rates, Melbourne, Victoria: Department of Labour.

Dr O'Brien is currently the Dean of the School of Economics at La Trobe University. He commenced working with computers at the University of Queensland in 1963 when he was a graduate student in mathematics. He then worked at the University of New England and the Australian National University, before joining the Economics Department at La Trobe University in 1972. He has worked at New York University and Warwick University. Current research interests are in economic dynamics and modelling and he has underway a large project modelling the socio-economic aspects of soil erosion in the Philippines. Teaching interests are wide with a special emphasis on Operations Research and Mathematical Economics. The research described here arose from his recent consulting work.

If it is useful then use it: Some experiences in the use of computers in the teaching of Chemistry.

Michael Page
School of Chemical Technology
South Australian Institute of Technology.

Twelve years ago I purchased some BASIC listings of drill and practice computer programs. This year I am involved in the production of an interactive videodisc. This paper describes some of the experiences and knowledge gained in the intervening years. It describes computer based education packages which have worked and some which have not, when C.B.E. has been useful and when it has been a disaster, which authoring languages have been found to be helpful and which have been a waste of time. Examples will be given of drill and practice, utilities, tutorials and simulations. And finally- has it all been worthwhile, and where do we go from here?

At a symposium reported in the Journal of Chemical Education entitled "Will Computers Replace Teacher Aids?, Professors?, Labs?, Should They?", participants were leaders in the field of computers in education. They alluded to the computer as a tool and as a new medium of instruction. There was general agreement that the computer provides new ways of approaching problems, but the consensus was that the computer is a tool and that good teachers cannot be replaced by a computer.

I first became interested in using computers as an aid to teaching about twelve years ago after a visit to Australia by Prof. Bitzer of PLATO fame. This, not surprisingly, coincided with the purchase of a CDC Cyber computer by the S.A. Institute of Technology making the use of a computer possible in the general teaching sphere of the Institute. Since then I have followed a tortuous path through macros, minis, micros, and a multitude of methods of authoring. I have been encouraged, discouraged, helped and obstructed. I have, as Marvin Minsky indicated to us two years ago, suffered the relearning process. At the end of it I realize we have come but a short way in the use of computers in the education scene. The information revolution is only just beginning and CD-ROMS, Interactive Videodiscs and satellite communications are opening up a whole new horizon of possibilities in education. Some of these possibilities we are aware of, but others we have no more knowledge of than the office clerk of the nineteenth century had knowledge of the word processor, database and spreadsheet. Just as they are tools of business so are similar packages tools for the teacher, and just as business and commerce proceeded with the abacus and steel knibbed pen so the teaching process should proceed with the tools that are available now. We cannot afford the luxury of waiting. Waiting for the ultimate teaching tool.

So this paper is not full of the latest in teaching technology which will turn the educational establishment on its ear. In any case education is too conservative for that. What it intends to do is to help take a little of the suffering and some of the drudgery out of the teaching process, and in so doing take some of the drudgery out of the learning process. It might even provide ideas to make students think (a revolution in itself!?) and provide time for the teacher to deal with individual student problems.

Philosophy

My philosophy as to whether to use computer software materials is based upon the following proposition.

Use it if it:

- enriches the course and if time is available
- saves the teacher time and doesn't disadvantage the student
- motivates the student to do well
- saves the student time by speeding up tasks or supplying information.

Purposely left out is the one of cost effectiveness. This is always a contentious issue. Are libraries, laboratories, photocopiers etc. cost effective as teaching tools or just traditionally desirable? I would question the purchase of a large computer system just for computer aided learning tutorials and drill and practice, but there are so many more things that can now be done with computers to make life easier for the student and educator. One must consider all these things if one is to consider cost effectiveness. Frankly I consider such an exercise as useless as trying to evaluate the cost effectiveness of a specialist book collection in a major reference library.

One last point in the philosophy of using computers in education is the fun of it. God knows one needs a sense of humour to be in the education profession at all and if using computers can contribute towards the outlet of that sense of humour so much the better. Learning should be fun it is the unlearning that causes the suffering and that comes later.

The Programs

a. Time savers -

1. Calculation checks

These have been used to date in two main areas. The first is to check students practical work. In our large first year practical classes one of the terrible time consumers for the instructor is "marking practicals". Student calculations are notoriously unreliable. So if one comes across a "wrong" result at the end of a practical report the questions arise; is it bad laboratory technique, use of the wrong "formula", or straight out arithmetic (calculator) incompetence? Since in my view the instructor is there mainly to teach correct laboratory technique time should not be wasted on the other two. So a suite of programs has been written to check the students' calculations. Such a calculation check is required before a report is submitted for assessment. The student is prompted to input his/her experimental values, and then the final value which the student has calculated.

The program then tells the student whether the calculation is correct to within the limits set by the instructor, is close to that value or is very wrong. It does not say whether the final value is correct in a practical sense. When I first introduced this over 50% of the students were making calculation errors at the first attempt. Of course they all blamed the computer until they recalculated and got it "right". By the end of the first term few errors were being made.

Not having to search for calculation, transposition or formula errors can reduce marking time by up to 50%.

One of the spin-offs of this was the discovery that students have a bad habit of rounding off intermediate values during a calculation. This can cause large errors in final values even when using calculators. So now advice is given early in the course on the inadvisability of rounding and the importance of the order in which computations are carried out. Those who do not listen suffer the consequences.

It is also noticed that students usually check their calculations in small groups, and errors are often picked up by their peers and corrected quickly, and at a time when the corrections are relevant to the task in hand and not say a week later when reports are marked or handed back.

The second area in which this technique has been used is for assignment checking. This has allowed more assignments to be set and marked resulting in better detection of specific weaknesses in first year students.

2. Individualized Assignments.
Again not a new idea, but a great time saver and student motivator. Sets of individualized numerical problems are produced using a random number generator. These are printed complete with an answer sheet for each student. A set of answers for each set of problems is printed for the instructor.

It takes little time to mark the answer sheets for each student and blind copying is eliminated. Peer help is encouraged since, firstly it saves the instructor time, and secondly, experience has shown that "lazy" students do not get the problems done for them. Also answers cannot be passed down from one year to the next, a practice common with students for generations!

Such techniques as the above have been used for some years at the University of Texas at Austin to "cope" with 800 first year chemistry students.

3. Drill and Practice.
Now once again finding favour with educationalists, drill and practice is used extensively as pretest sessions for a self paced first year Chemistry course. Each of the 20 units of the course requires, apart from the completion of an *assignment, a test to be passed based on the contents of a chapter of the recommended text. The pass mark is about 90%. The computer sessions include all the questions that a student is likely to be asked in a test. Incidentally no two tests are exactly the same and a student can repeat the testing process until mastery is obtained. Students soon realize that time spent using the programs saves much time later.

This emphasizes an important principle of the use of computers in education. The use of the computer must be seen to serve a useful purpose and not just "nice to do". In this case passing the test first time after proper preparation is a tangible goal with concrete benefits.

The tests themselves are not given at the computer terminal. The reason for this is that the taking of tests and the immediate marking of them gives an important contact between instructor and student, and allows subtle misconceptions about the subject matter to be dealt with at the time.

4. Utility Packages
Commercial programs such as word processors for reports, statistical packages for processing laboratory results etc. are being used more and more by students as they become convinced of their time saving attributes. One longs for the day when all reports, essays and assignments will be submitted perfectly typed with no spelling errors.

a. Simulations

These are used to illustrate theoretical concepts when they are first encountered so freeing laboratory work from being timetabled to fit in with lecture material. These simulations are not intended to be substitutes for work in the laboratory.

A simulation of enzyme kinetics is used as a follow up to a laboratory experiment where time constraints do not allow all the experimental variables to be investigated.

A simple nuclear chemistry simulation is used as a substitute for a laboratory experiment. In this case it is desirable that the students appreciate the unique laboratory principles, but facilities to mount the experiment are not available for reasons of cost.

Apart from the value of allowing students to apply theoretical knowledge (or lack of it) these simulations have highlighted the need for students to gain experience in planning their own experimental work. In most scientific laboratory sessions all the planning has been carried out by the instructor in order that the experiments should "work". In other words the student is left with a recipe to be followed slavishly without real understanding of the concepts. This delights the student and makes practical courses run smoothly, but leaves a gap. The computer simulation is an ideal vehicle for training in organization and planning as a wrong plan does not waste valuable materials and equipment and, since simulations produce results rapidly, time is also saved.

Simulations are also used as animated examples in lecture and tutorial classes allowing a freedom to "see what happens if", and to encourage student involvement.

There have been simulation packages which have not been successful. One reason for this has been that the simulation itself has been too close to reality and has wasted time in repeated animation when quick results was the objective. Again we come back to the timesaving aspect of computer based education.

c. Others

No mention has been made of computer generated tutorials. My experience so far is that these are resisted by the average student as a waste of time. They regrettably can see no point in doing a program because it "it is good for you". Of course those students do not read their text books and references for the same reasons, and only look at duplicated notes if they think an exam question is hiding in there somewhere.

We must not think that the use of computers in education is going to have a dramatic effect on the behaviour and attitude of students towards their own education. Of course a well produced piece of courseware presented at the right time in the right context will stimulate and motivate students, but no more than the same material well presented at the right time and in the right context by a teacher. The computer is still a tool and a tool which is rapidly losing its awe to the student population. In fact computers are getting a "bad press" because popularly they are being blamed more and more for the errors of their users.

So in general I only use computer generated tutorials for specific cases of difficulty, and even then find some resistance to their use.

Authoring Systems and Languages

At present I use PC-PILOT on an IBM-XT compatible microcomputer with a hard disc, and colour of course. I have had experience with several other authoring type systems namely STAFF and DAL on mainframes, and PILOT, ENBASIC, Q, QCAL, PAL and MICROTEXT on microcomputers.

The reason for choosing to use PC-PILOT is partly subjective as is the choice of authoring systems by most people. In my case I needed a system for the IBM microcomputers which are available for student use at the S.A.I.T. I also needed a system with good graphics, a choice of colours and fonts, good string handling routines and good mathematical capabilities. I was not so concerned with computer managed courses although PC-PILOT has this capability at an elementary level.

I was not interested in an automatic programming system as my previous experience with such facilities had shown them to be restricting in the long term and lacking in flexibility. Such "filling in the blanks" systems look very attractive to a newcomer to authoring, but unless one can break out of that stage easily one soon becomes frustrated with the restrictions imposed. Actually PC-PILOT does have such a facility with the advantage that it produces the source code which can be manipulated later as one's experience grows. After some two years working with PC-PILOT I have now developed a collection of screens and routines which I use extensively giving me considerable flexibility.

PC-PILOT is a system which uses a simple text file created with a very good screen text editor. This contrasts with other authoring systems which are menu driven. Which is the best system I think is a matter for the user to decide. I prefer the text file probably because I am used to it and can easily visualize the lesson screen from the text file screen. I have also used a ram-disc and now a hard disc to switch quickly from the text file editor to the PILOT driver and so see the lesson as it develops. I can also pull in from my library collection, screens and routines which develop the lesson rapidly and with few errors. These routines include such things as matching tests, multiple choice question screens, true/false routines etc. and a lesson shell which sets up special characters for chemical symbols, windows, timing and marking macros. So I have probably reached the point where to change to a new and possibly more efficient system would be resisted because of the mass of material already developed and the chore of learning a new routine.

I have tried to concentrate on the positive aspects of the choice of an authoring system, and do not wish to dwell on the negative aspects of other systems, since as I have said, the choice depends upon what you want to do and upon your personal preference for the type of system you wish to use which will be coloured by your previous experience.

It is important however that, when evaluating a system, one makes sure that it is the latest version as software development can be rapid, and a system which in its first form may be primitive can, in version two, be just what one is looking for. Evaluations of authoring systems could be one area for consideration by ASCILITE with perhaps evaluation copies of the software and manuals available for short term loan by members along with a list of user contacts. Such a service would be invaluable to the new author and the more experienced user looking for alternatives and updates.

And in the future ...

My personal future will be to write more and obtain more of the types of courseware which has worked in the past. New things to write not mentioned above are pre and post laboratory tutorials and tests, past exam paper help and tips,and a suite of support programs for disadvantaged students, and I have to program and test that videodisc we have just produced.

I shall also continue to buy, borrow (but never steal) as much CBE material as possible because this is infinitely faster than writing it. And if the material is not exactly as I would like it I am quite prepared to slightly change what I do to fit the material, and if the material is good it will not do any harm.

The future of computers in education is assured, but the future of computer based education or computer aided learning or computer assisted education or whatever it is called will continue to be a rocky one. My experience over the years has seen computer based education greeted with enthusiasm by a few, treated with apathy by many and met with opposition both open and subtle by a surprising number of educators. It is a pity that not enough teachers see computers as the tool for better education, and a tool to be mastered not ignored or thrown aside.

The rapid technological advances in the fields of information exchange and storage are about to have a dramatic effect on the way we live. A flood of videodiscs and compact disc storage devices are about to appear and educators should be ready to take advantage of the wealth of information soon to be at our fingertips. This technology will never replace the teacher, but if used with intelligence, will give the teacher the time and freedom to give students the quality of education too often missing in our Schools, Colleges and Universities.

> The immemorial problems of teaching endure but at least we have been vouchsafed a marvellous new tool with which to chip away the darkness.
> Derek Davenport.

References

Will Computers Replace TA's? Professors? Labs? Should They? (1984). *Journal of.Chemical.Education,*. 61,(1), 26-35.

Castlebury, S.J., Culp, G.H., Lagowski, J.J (1973). J*ournal of.Chemical.Education,*. *50*(7),469-472.

M Page is Senior Lecturer in the School of Chemical Technology and in charge of Associate Diploma Course in Applied Chemistry. Teaches General and Analytical Chemistry and his Research interest is mainly computers in education Community interests _Theatre (performing) and portrait photography.

Computer managed learning in electronics

D Parsons, School of Engineering
R Hunter, Division of External and Continuing Education
Darling Downs Institute of Advanced Education

The availability of computer managed learning material has been extended to students enrolled in Associate Diploma in Engineering course and, in particular to students enrolled in electronics. The CML component comprises test instruments, facility to report home-based practical exercises, and circuit construction simulations. Two modes of operation are available - interactive in study centres and home, and mail-in mode. Some evaluation of student reaction to the CML component has been made.

Acceptance of distance education models has added impetus for new approaches to teaching. This impetus, together with expanding availability of new technology, has generated new bases for the support of learning. The Darling Downs Institute of Advanced Education (DDIAE) commissioned its computer managed learning (CML) system in 1984. The system added a powerful component to the range of media available for the production of learning packages.

Computer Managed Learning.

The term computer managed learning is applied to embrace the facets of testing and instruction. The application has allowed the system to support not only assessment and instruction but to manage reports of laboratory exercises. Adoption of a narrower definition would have limited the application of the medium.

The system interfaces with three data bases - student records, assignments and CML content. The latter allows on-line or batch input of module content. The assignment data base contains unique identification of all current assignments, weighted and non-weighted, together with descriptors, dates and assessment weightings. All CML test suites form part of this data base. The student records interface allows updating of academic progress. The system itself stores responses to each test item, giving the opportunity for item analysis. Production output is in print and diskette formats.

Test suites are designed to incorporate items of multiple choice, exact answer, answer within a range, answer within a sequence, one of a number of answers, some of a number of answers formats or any combination of formats. A feature is the guidance of students depending upon their performance and the supply of feedback specific to student's response. The figure below details the basic structure of each test suite.

Students elect to use the system interactively using microcomputers and diskettes or via a mail-in mode which is processed on campus. Mail-in operation provides the same feedback for tests and experiment reporting as the interactive mode. The computer assisted instruction (CAI) packages are available for interactive use only. Student access to the interactive mode is located on-campus, at study centres and, to a limited degree, at home. Selection of the mode of operation is made at the beginning of each semester.

Operation at study centres and on-campus utilises three diskettes. A student diskette

Figure 1 : Key Elements of Computer Managed Learning

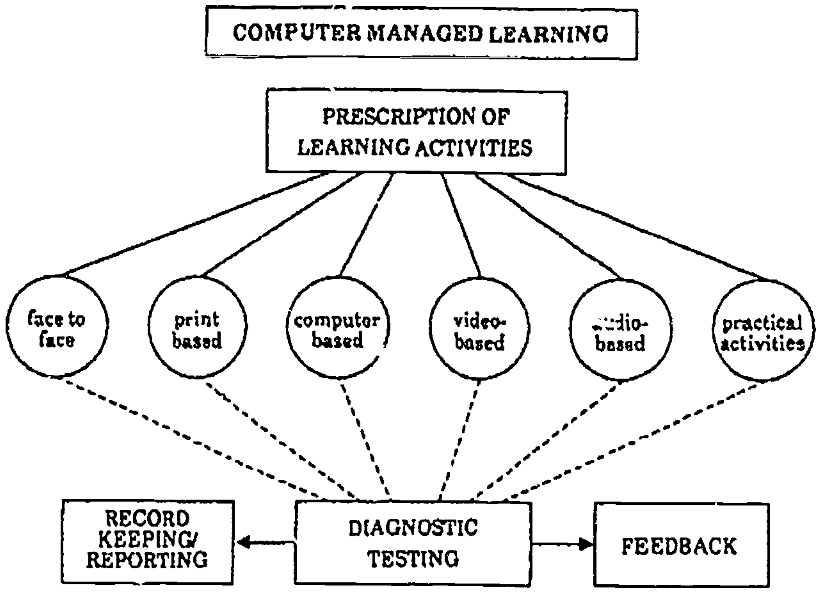
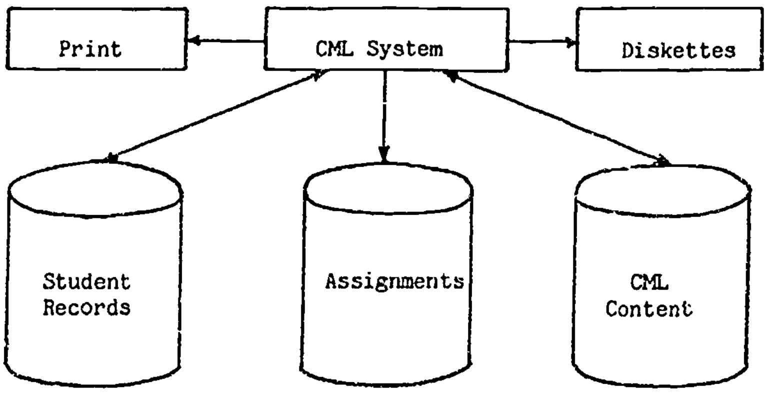


Figure 2 : CML Interface with Data Bases



which contains enrolment data, security, system access records and test results also contains the operating system. A content diskette holds test suites, experiment reporting formats and CAI material. A "mail" diskette has access and result data transferred to it and is used to update files on the student record base.

The CML system embodies fundamental principles. The medium is used as a management tool. The areas of instruction and testing are clearly identified. Progress is reported instantly; feedback is specific; remedial guidance is immediate. Within the constraints of assignment management self-pacing is possible. The remainder of this paper addresses the application of the principles to Electronics, in particular to the areas of testing, reporting and instruction through simulation.

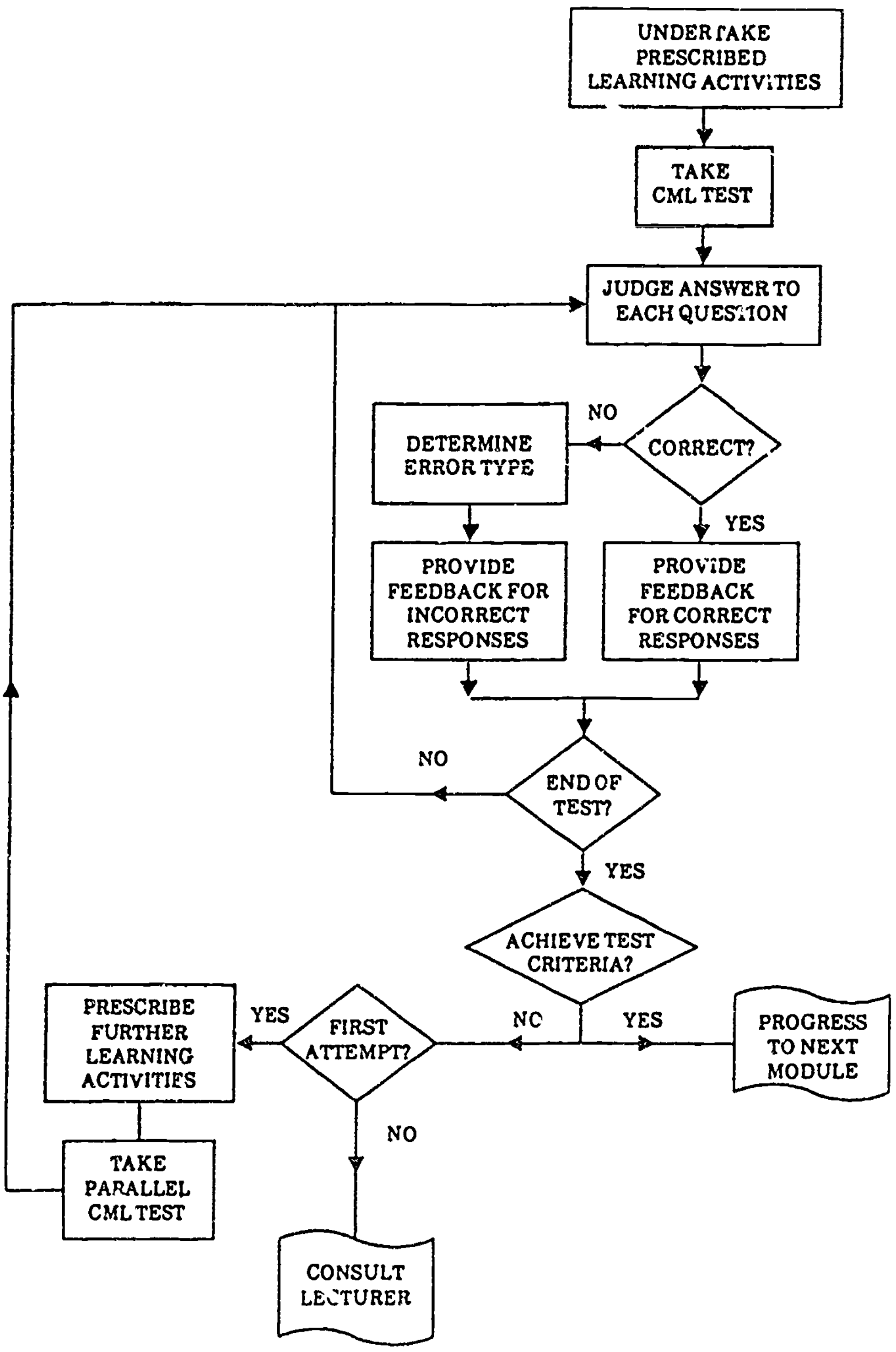
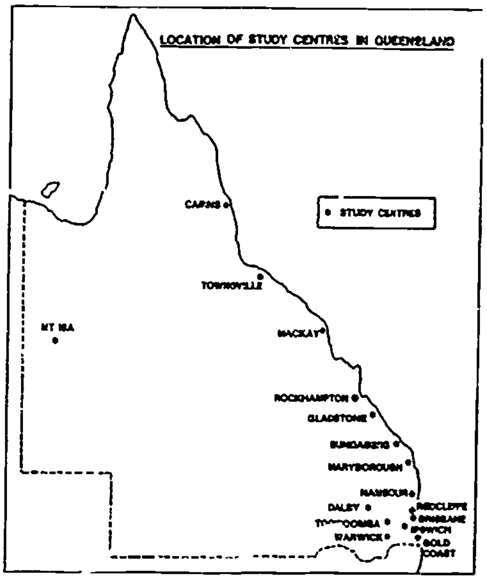

UNDERTAKE PRESCRIBED LEARNING ACTIVITIES
TAKE CML TEST
JUDGE ANSWER TO EACH QUESTION
CORRECT?
NO
DETERMINE ERROR TYPE
YES
PROVIDE FEEDBACK FOR INCORRECT RESPONSES
PROVIDE FEEDBACK FOR CORRECT RESPONSES
END OF TEST?
NO
YES
ACHIEVE TEST CRITERIA?
NO
YES
PROGRESS TO NEXT MODULE
FIRST ATTEMPT?
YES
PRESCRIBE FURTHER LEARNING ACTIVITIES
TAKE PARALLEL CML TEST
NO
CONSULT LECTURER

Figure 3 : CML System - Student Activity Interface

Figure 4 : Location of Study Centres



Testing

Testing is done by providing banks of questions, most of which are multiple-choice. About two-thirds of the questions are numerical, or calculation based, the rest being of a more descriptive nature. In electronics, a lot of reference must be made to graphics such as circuit diagrams and timing diagrams, and also to detailed data sheets such as those for integrated circuits. It is also important that students see these in their printed form, since this is the normal way they are available. So, even though it is possible to provide these on screen, students are referred to printed material. (This is also necessary for students without access to a computer).

Students require one to two hours of effort to do one of the twenty two tests provided. Three of these are nominated to be summative, with a total of twenty percent of the marks for the unit allocated to them. The remainder are optional. This approach was endorsed overwhelmingly by students who were questioned. They appreciated the fact that test questions covered every part of the work, and they did use the optional questions as preparation for examinations. It provided them with a means of self-assessment, and, in the case of the summative tests, forced them to work thoroughly in searching for answers.

In spite of the appreciation expressed by the students, they felt that the amount of feedback provided on incorrect answers was not as great as it could have been.

Some security against students using results from other students in previous years is given by providing several tests, and changing the one specified as summative from year-to-year.

Experiment Reporting.

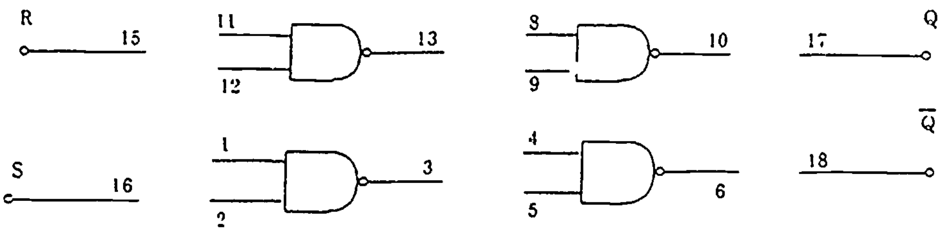
While many students are employed in situations in which equipment is available to them, there are some who are very limited by lack of such equipment. The electronics experiments have therefore been designed to be done on the kitchen table, with very simple and inexpensive equipment, typically just a power supply and simple multimeter. This system works very well and few students have difficulties, even though most are working alone.

The experiments are usually the assembly of a circuit and a series of measurements on, or changes to it. Very careful instructions are given to ensure that the circuits do work, and in ninety percent of cases, students have no trouble achieving basic circuit operation. The importance of doing something practical like this is in seeing it work and determining why, rather than just getting it to work. It is the understanding of the circuit and the principles behind it which are important, rather than the practical skills.

Eleven of these "home experiments" have been provided in one unit, and sixteen in another. They typically take one or two hours to do and in total are allocated up to thirty percent of the marks for the unit.

Figure 5 : A Design Problem

3 Design of true R-S Flip-Flop Draw here in the neatest possible way, avoiding crossovers as much as possible



2. Flip-flop design

Node number	1	2	3	4	5	6	7	
Connected to	□□	□□	□□	□□	□□	□□	□□	(2 20-2 33)
Node number	8	9	10	11	12	13		
Connected to	□□	□□	□□	□□	□□	□□		(2 34-2 45)

About half the students feel that they took too much time to do, but still wanted them to be a compulsory part of the work. Most students expressed some satisfaction with the practical familiarity gained with electronic components, and with the sheer experience of working with circuits.

Given the philosophy that doing successfully is the major objective, the marking procedure becomes (90% of the time) confirming a lot of correct numerical results. The use of a computer to mark in this case is essential to allow the experiments to be as extensive as possible. It also allows of course detailed feedback comments to be made, based for example on specific results.

There are at least two aspects of electronics where the ability of the computer to do calculations on students results is particularly helpful.

1. The performance of electronic devices varies greatly from one unit to the next, (and from one student to the next due to differences in power supply etc). Correct operation of the circuit can be determined by observing correct trends in results rather than absolutely correct numbers.

2. It is useful sometimes to have a "design your own" approach. The results can vary widely and correct design can be determined by analysis of reported design values. Figure 5 shows an example of this.

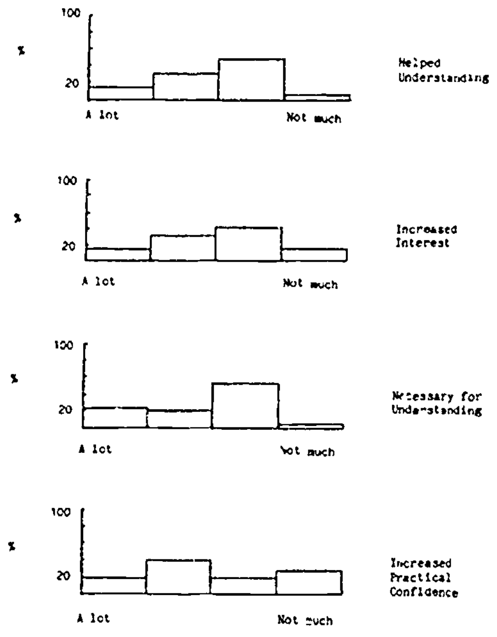
Because understanding is the goal, questions are attached to each experiment which call for analysis of the work just done at a practical level, and interpretation of results. Appendix 5 is an example of such a question.

Student reaction was sought to the experiments to determine attitudes about the extent to which doing the experiments:

- helped understand the material in the unit
- increased their interest in the subject
- were necessary for a good understanding
- increased confidence in handling electronics.

Figure 6 shows a distribution of the opinions offered in each case.

Figure 6 : Attitudes to Experiments



Simulations.

These are an attempt to overcome some of the limitations of the equipment shortage at home. They also encourage a "try it and see what happens" approach to the practical experimentation of the kitchen table. If these practical experiments do not work, no-one can tell the student why. With the simulations, the computer indicates what is wrong.

For the range of circuit types (using diodes, transistors and operational amplifiers) the student can do the following on the screen of a study centre computer.

1. Analyse a selected circuit with values provided which do produce a working circuit.

2. Design a circuit by selecting a particular arrangement, and the values of components. Totally incorrect values are prohibited by the system. A badly-designed circuit which still works is allowed so that the analysis will reveal its limitations.

3. Any of the above circuits may be altered and again analysed.

More than half the students trying this purely optional exercise lived over 10 km from a study centre, and most used the system several times. They felt that the exercises did help them to understand the circuits, and to become familiar with standard values of components. The writers personal impressions are that there are some people who respond to this kind of activity, and some who do not. We will be in a better position to assess the value of simulations when more students have easier access to a computer.

Impressions

Presented are some personal observations. The effort by staff into these computer-based facilities is huge. People involved were lecturer, instructional designer, administrator, programmer and production staff. Of these, the lecturer is the only one who hopes to receive a direct benefit in terms of reduced working load. Even so, with the typical numbers of students concerned (70-90) the lecturer would feel the need to run the material largely unaltered for several years.

The major benefit is to the student who has many more opportunities for formative exercises, receives work back more quickly and receives much more detailed comments on his work. There is some anecdotal evidence that more students are spending time ringing lecturers directly with specific questions about work given.

References.

Parsons, D.J. (1983). Practical Work in the External Training of Electrical Engineering Associates in Queensland, Australia. *Int. J. Elect. Eng Educ.*, *20*, 101-110.

Parsons, D.J. (1983). The Value of Practical Work in Teaching Electronics to External Students. Australian and South Pacific External Studies Association, 6th Biennial Forum.

David Parsons is a lecturer in electrical engineering at the Darling Downs Institute of Advanced Education in Toowoomba, Queensland, where he teaches electronics. A major part of his teaching is done externally, and has involved developing home-based teaching materials and strategies.

Bob Hunter is Associate Head (Resources) of the Division of External and Continuing Education at the Darling Downs Institute of Advanced Education. He is an instructional designer working on the development of computer managed learning.

Formative evaluation systems for CAI

John A. Price,
Accounting and Business Law,
University of Melbourne,

A major problem with CAI systems is the quality of their instructional materials. CAI is an electronic form of teaching and the normal opportunities for teacher observation and feedback are not available. A formative evaluation module can be included in a CAI system to help to overcome this. It requires little in the way of additional programming work to incorporate a formative evaluation module into a CAI system Yet CAI authors have been slow to recognize this potential. The result has been the use of multiple-choice questions in CAI which are not adequately tested. The author describes a CAI system that he has developed which includes a formative evaluation module. The system runs on an IBM-PC.

Two phases in the evaluation of educational material and processes are commonly recognized. These are: formative evaluation (i.e. while a CAI system is being developed) and; summative evaluations (i.e. when the CAI systems is completed). Formative evaluation rather than summative evaluation is used in the development and updating of instructional material. The author designed and implemented a CAI system which incorporates a formative evaluation module. In this paper the principles of formative evaluation and its application to CAI are discussed. Formative evaluation is discussed in the first section of this paper.

Formative evaluation

Bloom, Hastings & Madaus (1971), the authors of a major work on the evaluation of learning and teaching processes, note that:

> for the most part the teacher-made tests are summarized to show the score or marks of individual students. Only rarely does the teacher use them as the basis for modifying instruction.(p.135)

Student marks are translated into grades and used to separate students into the more and less able, so that the more able may progress to the next stage or to a higher stream. However data on student achievement has another valuable educational application. It can provide the basis for improving instruction. Bloom et al. (1971) urge educators to use the student-tests as the basis of "quality control" of instruction and not: "as it is most frequently used in the existing educational systems ... the grading and classification of students" (Bloom et al. 1971, p.7). This view of evaluation which considers student grading leads educators to make decisions about students rather than about teaching material and methods. Decision making in the educational process must do more than grade students. This view is supported by Kandaswamy (1980) who contends that it must involve:

> making decisions about which components of the training material are to be modified in order to make the materials instructionally and motivationally stronger. (p.19)

Programme developers determine the effectiveness of their material at various stages in its creation. During programme development the CAI system is used and the responses to the multiple-choice questions are recorded. The information could be used in a step-wise development of the programme. In step one it could be used to determine the validity of the test questions and in step two, to assess the strengths and weaknesses in the preceding instruction.

The means of providing information to students, teachers and the instructional designers so as to improve the teaching/learning process is called 'formative evaluation'. This term was first used by Scriven (1967) in a paper on the methodology of evaluation. Bloom et al. (1971) note that:

> it is [Scriven's] view that formative evaluation involves the collection of appropriate evidence 'during' the construction and trying out of new curriculum in such a way that revisions to the curriculum can be based on this evidence. (p.117)

In formative evaluation data on student progress and learning is used to determine the efficacy of the instructional materials and teaching processes. Formative evaluation is an important tool used in the trial implementation of a new programme at each stage in its development. Bloom et al. (1971) note that in the application of formative evaluation:

> it may be determined at each step in [this] process whether the process is effective or not, and if not, what changes must be made to ensure its effectiveness. (p.8)

Measuring student progress, such as test results, provides information that is used to improve the instructional material or processes. It involves tests which are as much a test of the programme as the ·re of the student. As Weiss (1977) notes:

> formative evaluation produces information that is fed back during the development of a curriculum to improve it. It serves the needs of developers. (p.17)

The information indicates to teachers and instructional designers the need to improve the teaching material or process. Areas of weakness, such as lack of understanding or practical ability, are identified. A teacher modifies his/her approach while a lesson is taking place based on student feedback. One of the limitations with an electronic teacher such as CAI is that teacher observation is limited. However this can be overcome by incorporating a formative evaluation module into a CAI system. A CAI formative evaluation module can:

- assist the initial design and creation stages;
- improve the instructional content;
- check the logical sequence;
- help to select questions for CAI drill and;
- improve the communication aspects.

The improvement of the drill component is the first stage in the application of formative evaluation to CAI. This aspect of improvement of CAI is the subject of this paper. In the discussion to follow, the CAI system and the functions of the recording and formative evaluation modules are described. This corresponds with the top down design methodology used in the construction of the CAI system for this project (Price, 1985; Price, 1987).

The CAI system

CAI is a method of teaching which uses a computer to provide instructional material and interacts by asking the student questions. CAI tutorials are characterized by the use of multiple-choice questions. A CAI tutorial instructs the student and asks a question on the material just presented. For this reason they are frequently described as 'instruct and drill' systems. When students respond they are provided with immediate feedback as to whether their answer is right or wrong. If they are wrong, they may be provided with hints as to the correct answer, encouragement and so forth. The communications between the student and the CAI system are controlled by the system interface. This interface also allows communication with other users, such as program authors, teachers, mentors and administrators. All these may at some time need to access the CAI system to obtain student information or update programmes. In the CAI system to be described in this paper there are two types of users. These are teachers/administrators and students. The system menu provides a route for each user to access the CAI system and is illustrated in Figure 1. The CAI system controls access to data in the CAI system by a hierarchy of privilege. This is an important consideration because of the need to protect the integrity of student records and teaching material.

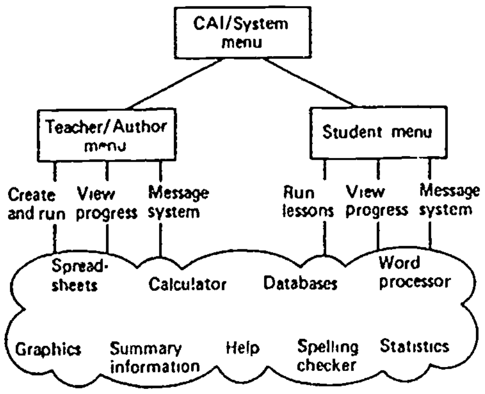


Figure 1. CAI system menu.

Typically the teachers' menu allows the creation and editing of tutorials and viewing student progress, while students can only run tutorials and view their own progress. The ability to provide progress reports to teachers and students depends on the CAI systems recording module. This is discussed in the next section.

The recording module

Gerhold (1980) found only a small amount of interest shown by authors in keeping user statistics in more than six years of CAI use at the University of Western Washington. If records are kept then they should be useful, and not just record that a particular student has finished a lesson or topic. The simplest form of recording system is when the number of correct responses is recorded against the students name. An example of a CAI system which includes a simple recording module is illustrated in Figure 2.

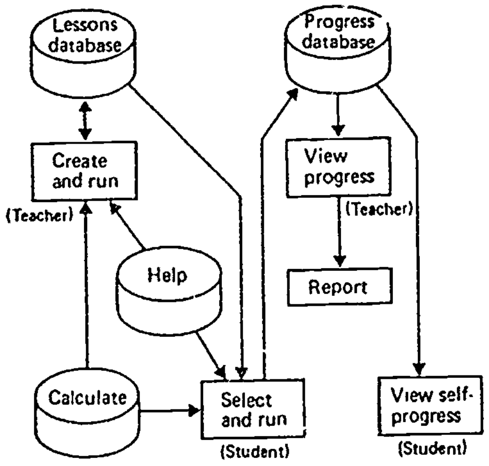


Figure 2. CAI design with a simple recording system (based on the "progress" database of McDevitt & Price 1984).

To implement a formative evaluation module the recording module must store more than just the on-line information provided to students, such as the number of correct responses. It is necessary to store the student ID and that student's answer to each question on the first and second attempt. The author developed a database that held this additional information. It has more attributes (fields) than the earlier database recording system developed by McDevitt & Price (1984). CAI systems have a variety of question types such as multiple-choice, true/false or fill-the-gap. The database is sufficiently flexible to cater for most types of question. The author has named this new database the "extended" database. It stores the student's responses to each question on the first and second attempt. The teacher/ administrator has access to the data and can process it on a weekly/ monthly basis with a formative evaluation module. The formative evaluation module is discussed in the paragraphs to follow.

Formative evaluation module

The writers of multiple-choice questions should consult a text such as Thyne (1964) for general advice on test construction. However Allen & Yen (1979) contend that:

> anyone who develops and uses tests should {also} be familiar with the procedures for test development and evaluation. (p.142)

This does not mean that teachers need to understand mathematical statistics. It means that they can use statistical measures in the selection of multiple-choice questions or "items" (as they are called in the measurement literature). The formative evaluation module of the CAI system developed for this project is an item-analysis program.

Item-analysis techniques and computer programs (e.g. CSHE, 1979) have been developed to test multiple-choice questions used in examinations and tests where students are being assessed (Nedelski, 1965). In CAI the quality of the drill component is still important even though students are not being assessed. The technique of item-analysis provides information that is used in the formative evaluation of the drill component of CAI. An item-analysis program identifies questions in a CAI system that are ineffective.

A multiple choice question is ineffective for any of the following reasons:

a. its too (easy) obvious and a high proportion are correct;
b. its too (hard) difficult and a high proportion are incorrect;
c. both good and poor respondents score equally indicating uncertainty and probably ambiguity or no exact correct response;
d. negative correlation, suggesting answer supplied by the item's author may actually be wrong and;
e. if one or more of the distractors attracts very few correct responses indicating that it is not working and is probably implausible.

An item analysis print-out is illustrated in Figure 3 and shows thr items from a recent batch of CAI drill. There are two important statistics shown in the item analysis print-out. These are the item difficulty and the item biserial correlation. An item difficulty close to 1.00 indicates an easy question. Conversely an item difficulty close to zero indicates a very difficult question, which few students 'get right'. The item biserial correlation gives the correlation between performance on an item (question) and on all other items (questions) in the drill.

Below the author demonstrates the application of item-analysis statistics for the three items. Teachers and test-developers can successfully use item-analysis without needing a full understanding of statistics.

- Item 1 is too easy. Nearly all the students were correct on this item. However there is a good correlation between performance on this item and the rest of the drill.

- Item 6 is a poor question and should be modified or dropped. The .48 item difficulty is quite acceptable. However the biserial correlation of .187 indicated that there is a low correlation between performance on this question and performance in the remainder of the drill.

- Item 22 is a good question. Both of the level of difficulty and the biserial correlation statistical measures are satisfactory. This question should be retained.

There is a need for at least some easy questions in a CAI drill. This is near the beginning of a lesson where they serve to give students confidence. Item 1 is clearly such a question.

Concluding remarks

Microcomputers have increased the opportunities for the creation of CAI (e.g. Price, in press) and improved authoring systems have made program creation and modification easier. However as CAI authors become more adventurous they need tools which will improve the content of their material. Formative evaluation is one such tool.

Figure 3: Item analysis of three multiple-choice question.

```
+-------------------------------------------------------------+
|                                                             |
| ITEM 1   difficulty - .920    1      2      3      4      5 |
| number of responses - 88      1      2      4     81      0 |
| top score        UPPER 1      0      0      0     17      0 |
|                  UPPER 2      0      0      0     18      0 |
|                  UPPER 3      0      0      0     18      0 |
|                  UPPER 4      1      0      0     16      0 |
| low score        UPPER 5      0      2      4     12      0 |
|        TEST SCORE MEANS    23.0   15.5   16.3   30.4      0 |
|        CONV SCORE MEANS     9.9    6.5    6.5   12.8      0 |
|        Biserial correlation - .88                           |
|                                                             |
|-------------------------------------------------------------|
|                                                             |
| ITEM 6   difficulty  - .489   1      2      3      4      5 |
| number of responses  - 88     6     15     24     43      0 |
| top score        UPPER 1      1      0      4     12      0 |
|                  UPPER 2      1      3      6      8      0 |
|                  UPPER 3      0      4      6      8      0 |
|                  UPPER 4      1      4      4      8      0 |
| low score        UPPER 5      3      4      4      7      0 |
|        TEST SCORE MEANS    26.2   26.6   29 8   30.4      0 |
|        CONV SCORE MEANS    10.9   11.6   12.8   12.8      0 |
|          Biserial correlation - .187                        |
|                                                             |
+-------------------------------------------------------------+
|                                                             |
| ITEM 22  difficulty - .682    1      2      3      4      5 |
| number of responses - 88      6     60      5     13      3 |
| top score        UPPER 1      1     15      0      1      0 |
|                  UPPER 2      0     17      0      1      0 |
|                  UPPER 3      0     15      1      2      0 |
|                  UPPER 4      3      9      0      3      2 |
| low score        UPPER 5      2      4      4      6      1 |
|        TEST SCORE MEANS    24.0   32.0   20 6   24.5   23.7 |
|        CONV SCORE MEANS    10.2   13.9    9 2   10.4   10 2 |
|          Biserial Correlation -   690                       |
|                                                             |
+-------------------------------------------------------------+
```

References

Allen,M.J. & Yen,W.M. (1979). *Introduction to measurement theory.* Monterey,Calfornia: Brooks/Cole Publishing.

CSHE (1979). *Testan users guide: Departmental handout.* The Centre for the Study of Higher Education, University of Melbourne.

Bloom,B.S., Hastings,J.T. & Madaus,G.F. (1971). *Handbook of formative and summative evaluation of student learning.* New York: McGraw-Hill.

Kandaswamy,S. (1980). Evaluation of instructional material: A synthesis of models and methods. *Educational technology,* June, 19-25.

McDevitt,A.A. & Price,J.A. (1984). dBase II: A tool for Computer Aided Learning. *Proceedings of the Second Conference on Computer Aided Learning in Tertiary Education,* Brisbane, 167-187.

Nedelsky,L. (1965). *Science teaching and testing.* Chicago: Harcourt, Brace & World, Inc.

Price,J.A. (1985). Software engineering: A guide for CAI. In J.A.Bowden & S.Lichtenstein (eds.), *Student Control of Learning Computers in Tertiary Education* (pp.35-60). Melbourne: IBM.

Price,J.A. (1987). Teacher produced CAI using software engineering. *Collegiate Microcomputer,* 5(3), 239-243.

Price,J.A. (in press). Developing computer aided instruction systems for microcomputers. *International Association for Computing in Education Journal* (formerly AEDS Journal).

Scriven, M. (1967). The methodology of evaluation. In R.Tyler, R.M.Gagne & M.Scriven, *Perspectives in curriculum evaluation.* AERA Monograph series on curriculum evaluation, No.1. Chicago: Rand McNally.

Thyne,J.M. (1964). *Principles of examining.* London: University of London Press.

Weiss,C.H. (1972). *Evaluation research: Methods for assessing program effectiveness.* Englewood Cliffs, N.J.: Prentice-Hall.

John A. Price known as Tony is on the staff of the Accounting and Business Law Department of the University of Melbourne where he is involved in computing literacy training. He is also in the process of writing up a thesis for the M.Ed. degree of the University of Melbourne. His studies at the Centre for the Study of Higher Education, the University of Melbourne relate to the use of instructional video in computing literacy training. He has been involved in computer training and the production of CAI systems at the University of Queensland and for the Royal Melbourne Institute of Technology. He has publications in computer education journals such as Collegiate Microcomputer and the AEDS Journal and the Journal of the South-Western University of Finance and Economics, Peoples Republic of China. Originally form Wales (U.K.) he has Bachelor of Science degree from the University of Wales and a Masters in Economics degree from the University of Queensland.

Automated Storyboards — a plausible authoring aid?

S. Sampath and A.J. Quaine
Department of Computer Science
Australian Defence Force Academy

The effectiveness of an educational package generated by use of any authoring system depends on the methodical development of the instructional design. There are number of stages involved in a complete instructional design. One is the development of the detailed instructional material, as a series of instructional events to be presented as a sequence of frames. The most common approach for a systematic development of this material is to prepare the detailed layout of each frame on paper. These layouts are traditionally referred as storyboards. The data entered via electronically prepared storyboards has been used [8], to author lessons using the WISE authoring system. Extending this idea, this paper surveys its application to other authoring systems and suggests that storyboards can be generalised as a distinct data type to be processed electronically on a wider environment, and so provide a general tool for effective instructional design. The development of such tools offers promise of efficiency in the authoring cycle, and further author support through electronically accessible high-level documentation.

Systematic development of instructional design is the most important part of producing Computer Based Instruction (CBI) material. Basic development stages of an instructional design system for any computer based courseware can be summarised as follows:

1. Identify the objectives of the subject matter to be taught according to the target population with the help of experts

2. Establish the hierarchy of topics to be presented according to the set objectives

3. Prepare the detailed layout of the instructional materials on paper according to the sequence in which the subject matter is to be presented

4. Prepare the material for presentation using an authoring system. (Romiszowski, 1986).

There are a number of steps involved in the development cycle (Romiszowski, 1986; Bork, 1984) for each of these stages, and it is important that each step be reviewed for the effective production of the educational package.

It is possible with a number of authoring systems, for the instructional designer to document the objectives and the topics of the subject matter electronically. Systems such as PCD3 (Plato's courseware design system for PC'S) and Course of Action (courseware design system for the Macintosh™ personal computer) allow the author to build a logic map of each section of the lesson, thus providing a very useful authoring tool. The third stage of the instructional design very much depends on the authoring system being used. Most experienced instructional designers, authors, and authoring system developers suggest the storyboard approach to pr pare the detailed layout on paper for e n frame or event according to the sequence in which it is to be presented. The layout of these storyboard forms depends upon the authoring system (reference note 1). Having prepared these storyboard forms, it would be ideal if the majority of the information entry could be passed on to a typist for straight forward data entry. Since most of the authoring system requires a certain level of expertise in using the software, it is not easy for a novice user to enter the data.

We have discussed in a previous paper (Quaine & Sampath, 1986) how the data entered via electronically prepared storyboard forms has been used to author lessons using the WISE authoring system. The aim of this presentation is to extend this idea to other authoring systems and suggest a unique storyboard form that can be adapted to any authoring system. This will provide a general authoring tool for effective instructional design and offers promise of further efficiency in the authoring cycle.

Authoring efficiency vs Authoring tools

The quality of CBI material depends on a variety of factors. However, as mentioned by Anver *et al* (1984), the three major factors contributing to the quality of production are:

1. Experience with instructional design
2. Experience with authoring software
3. Experience with subject matter.

Of these factors they have shown that experience with authoring software provides the most cost effective method of improving authoring productivity. This finding concurs with the survey done by Lommel (1986). However, the provision of tools like automated storyboards as an additional authoring aid can reduce the need for expertise in using authoring software. Table 1 shows the results of our survey of different authoring systems, the authoring time required to prepare one hour of courseware after the design and analysis of the instruction materials.

It is clear from the table that development time of the CBI courseware is highly variable and depends on the complexity of the course being developed. The courseware prepared by AIP and Telecom involved more complexity than those prepared by Expert Solutions. However, the time quoted by Telecom seems high. The following reasons could be contributing factors to the magnitude of the time required.

1. Setting up the screens with responses and feedback is quite a difficult task using WISE. To obtain the required effect, the free response frames must be combined appropriately with calculation frames.

2. Setting up calculation frames in WISE requires special programming language skills, similar to those required for programming in Pascal.

3. The user design did not specify the preparation of detailed layout of the frames on storyboard.

The authoring time in this case could be considerably reduced if the front end software tools developed for WISE (Quaine & Sampath, 1986) had been used. AIP authors quoted that, in the complete development cycle of the courseware, starting from design and analysis, 20% of the development time is spent on creating relevant graphic images. The level of complexity of the courseware pr[illegible]d by Expert Solution using PCD3 and WISE was comparable. However, the authoring time using PCD3 is almost 30% less than using WISE. Probably the availability of additional tools in the content and strategy editors of PCD3, could be the reason for the reduced authoring time. Macleod (1983) states that the total development time varies from 100-300 hours (60-180 hours for preparing script and 40-120 hours of programming) for test only courseware to simulation type. An entire semester's chemistry course was developed at the university of Delware

AUTHORING SYSTEM	USERS	COURSE NAME	AUTHORING TIME HRS	TOTAL DESIGN HRS	DETAILED SCREEN LAYOUT
PAL Prologic	University of Queensland	Simple linear courses	70	NA	None
PCD3	Expert Solutions Sydney	Project Management	80	200	Storyboard
WISE	Expert Solutions Sydney	Using Community Services	120	260	Storyboard
PC CLASS	Dept of Army	Electronics Training	100	240	Storyboard
CourseMaster	AIP Melbourne	The Electricity Project	200	500	Storyboard
WISE	Telecom CBT branch Melbourne	Form Filling	400	NA	None

Table 1.

(Garton *et al*, 1984). They quote a total development time of 350 hours (105 hours for scripting, 195 hours programming and 50 hours of testing) for an half hour lesson. These results indicate that any additional software used as an authoring aid can produce cost effectiveness. From this survey we do feel that a general purpose storyboard form can be a distinct data type, that can be introduced for preparation of CBI materials, and processed electronically in the development of the overall pedagogical design of instructional material.

General purpose storyboard forms

The storyboard forms used in our previous paper (Quaine & Sampath, 1986) were specifically designed for WISE authoring system. Those forms cannot be directly used for other systems because of dissimilarities at the user interface. Neither do they use the frame name as the basic building block. Some do use named screens, or pages, which are similar to frames. There is a new breed of authoring systems that use events as building blocks, and each frame can be considered to consist of number of events or may be a single event. Merrill (1987) argues that the frame is a passive concept, and proposes a new terminology called an instructional transaction, and claims that a transaction is an active concept. However, the concept of the frame was earlier thought to be the correct model (Merrill, 1985). The concept of the presentation screen is clearly the unit of interest to the student, but from the author's point of view the display details could be a collection of frames, screens, pages or events. However, the key issue is that the material needs to be prepared and documented in the form of a storyboard. Table 2 shows the basic unit of current authoring systems.

All these systems provide the facility to set up different types of testing using the question styles like single or multiple word, free response, true/false, match the list, and multiple choice. The systems *Author* and

Authoring System	Basic unit for presentation	Menu, Language or Both
AIS-II	Frame - template	Menu
AUTHOR	Frame - template	Langauge
CourseMaster	Event	Both
Course of Action	Screen	ICON oriented
PCD3	Event	Menu
PC CLASS	Page	Menu
PLATO	Screen	Langauge
Prologic -PAL	Page	Both
SAM	Page	Menu
WISE	Frame	Menu

Table 2

CourseMaster allow the author to set up mathematical question types.

This summary is supported by Data Training's (1986) Annual Survey of CBT authoring systems, which suggests that the basic building blocks of the instructional material can be classified into three types :

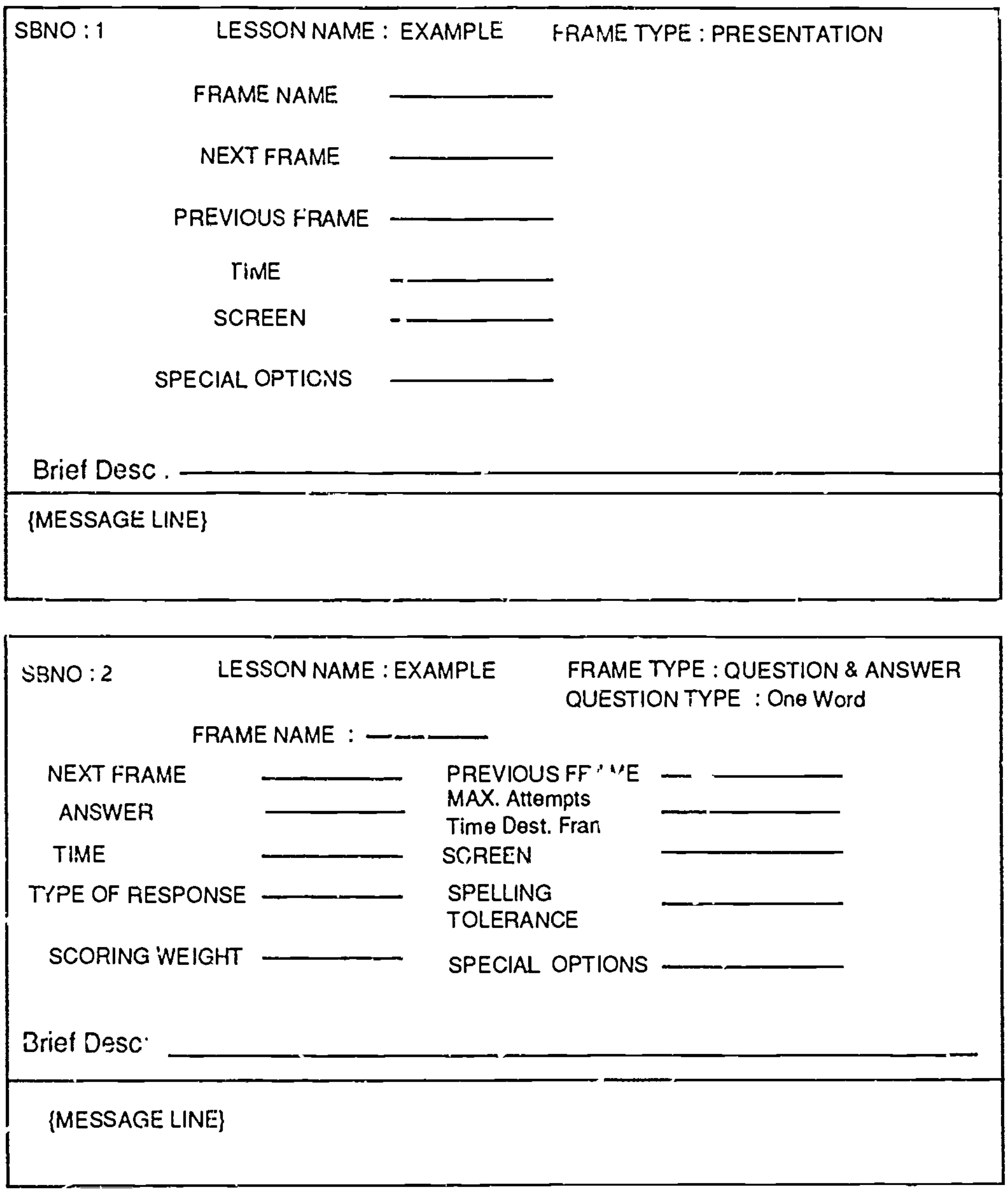
1. *Menu type.* Allows the student to select a particular section of the lesson.
2. *Presentation type.* One that does not require any response from the student.
3. *Question & answer type.* According to the student's response a particular action is taken.

The question & answer types can be any of the following types:

1. Single word response
2. True or False response
3. Multiple choice response
4. Free text response
5. Give responses by matching a list .

Other common features of these authoring systems are that they allow the author to give feedback for the student's responses and to have complete control of the student's progress through the lesson. So it is not difficult to extend the idea of processing the storyboards electronically, which we used for WISE authoring system. The storyboard processing of this system consisted of three processes:

1. Createsbs — to create storyboards.
2. Lessonoverview — to produce data flow of lesson logic
3. Printsbs — prints the detailed layout of storyboards.



SBNO : 1 LESSON NAME : EXAMPLE FRAME TYPE : PRESENTATION

FRAME NAME ____________

NEXT FRAME ____________

PREVIOUS FRAME ____________

TIME ____________

SCREEN ____________

SPECIAL OPTIONS ____________

Brief Desc . ____

{MESSAGE LINE}

SBNO : 2 LESSON NAME : EXAMPLE FRAME TYPE : QUESTION & ANSWER
QUESTION TYPE : One Word

FRAME NAME : ________

NEXT FRAME ____________ PREVIOUS FF ' 'E ____________

ANSWER ____________ MAX. Attempts
Time Dest. Fran ____________

TIME ____________ SCREEN ____________

TYPE OF RESPONSE ____________ SPELLING
TOLERANCE ____________

SCORING WEIGHT ____________ SPECIAL OPTIONS ____________

Brief Desc· ____

{MESSAGE LINE}

Figure 1: Storyboard part 1

The only process that has to be changed to make this system adaptable to other systems is the Createsbs, in order to include the different features offered by the other systems. Three levels of menu selections could be allowed in this process. The first level will be to establish whether the authoring system to be used is frame based or event based. The second level of menu selection will be to choose one of three basic types we have mentioned earlier i.e menu, presentation, question & answer types or any other type (other type could be special options provided by other systems). If the user chooses the question & answer type then it will display the third level of the

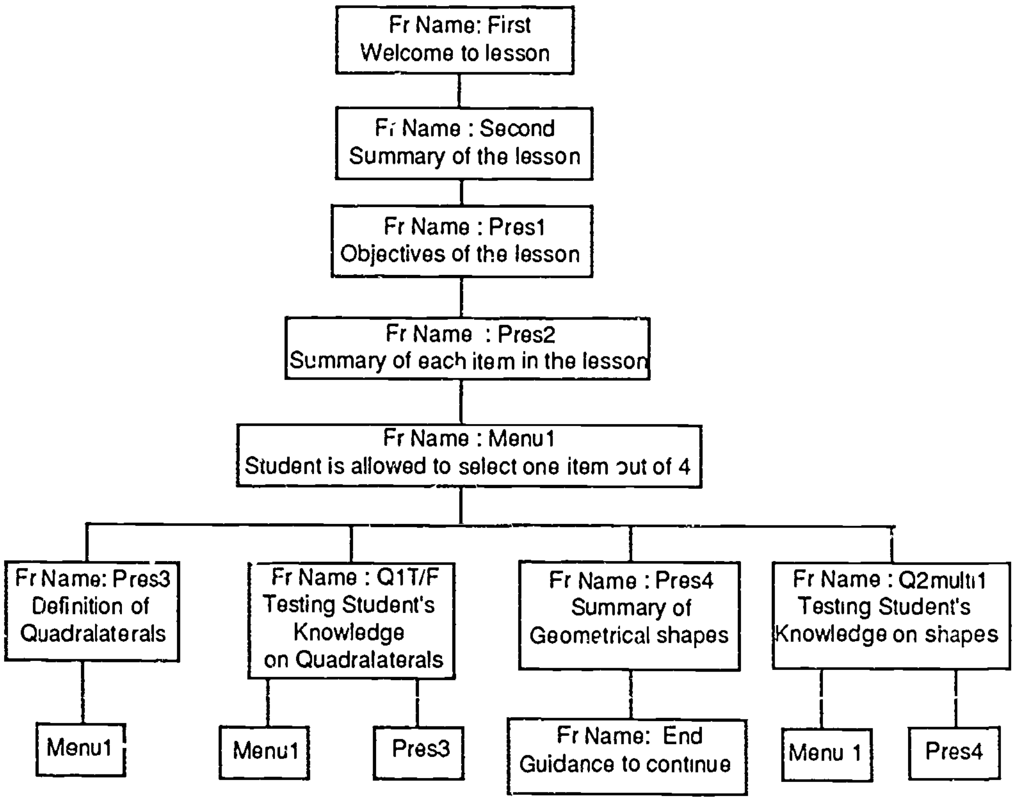


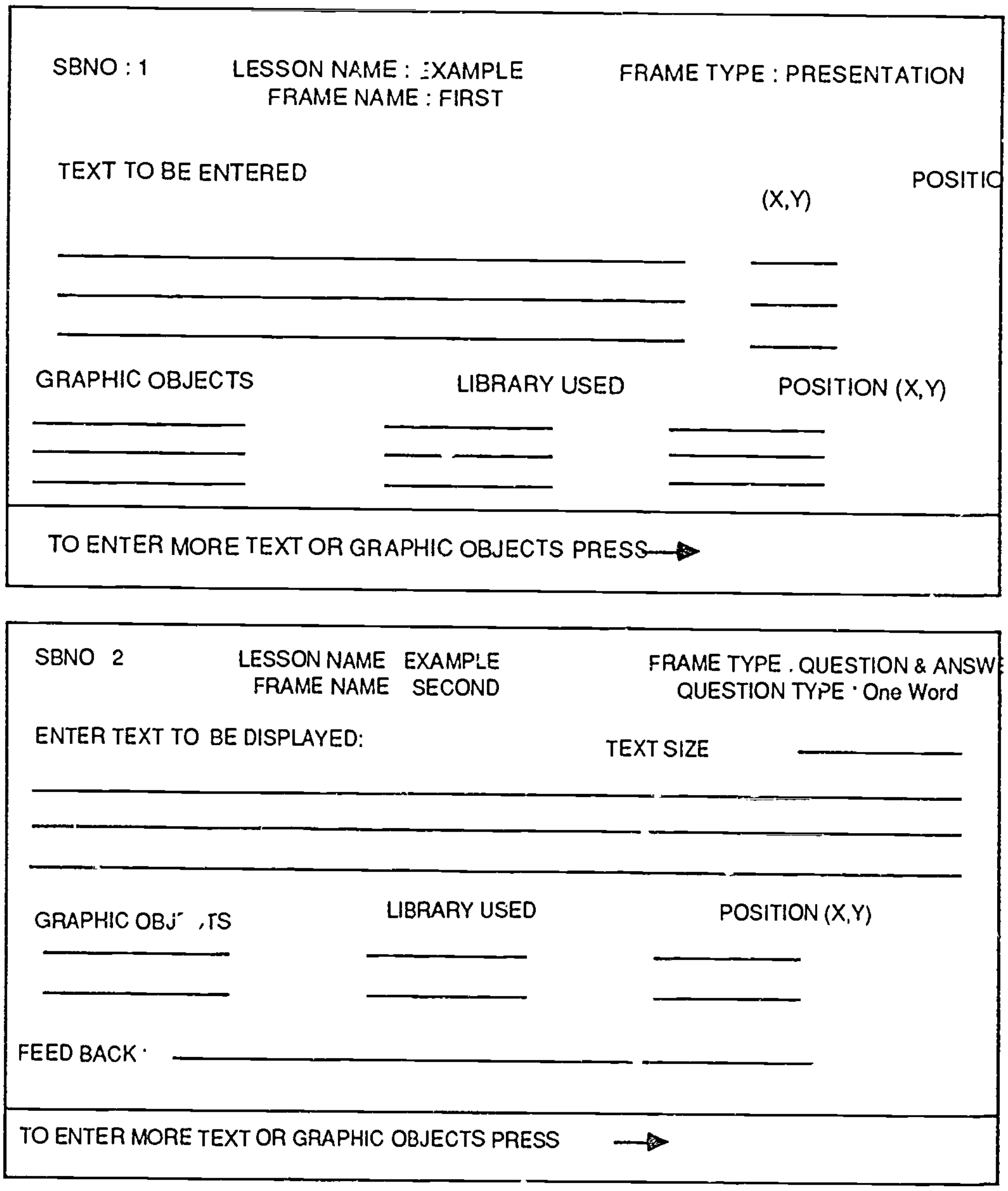
Figure 2: Logic flow of a lesson

menu which will allow the user to choose the type of question he wants to create. Figure 1 shows part 1 of the modified storyboard form for a presentation frame and one word answer and question type (for event type of authoring system the word frame will be replaced by event). Similarly other storyboard forms could be designed for other frame or event types.

The field Brief Desc in part 1 of the storyboard form is to allow the user to enter brief description of what is to be displayed in the frame. This description will be printed for each frame when the process Lessonoverview is run.

The output of Createsbs can be input to Lesson overview to produce the layout of the lesson as shown in Figure 2.

Except in the case of PCD3 authoring system, all the other systems provide the facility to create a set of graphic libraries and these can be called in any lesson. Most of these systems can indicate the exact location of the cursor and thus any graphic object from any library can be positioned at that coordinate on the display frame. AIP employs special graphic designers to create the graphic libraries required for the courseware. The CourseMaster authoring system provides some of the facilities of the storyboard processing we have developed for WISE. Their system can also produce the hard copy of the flow of lesson details in the textual format. This electronically accessible documentation is within the authoring system. The method of storyboard processing that is being suggested in this paper is independent of any authoring



SBNO : 1 LESSON NAME : EXAMPLE FRAME TYPE : PRESENTATION
FRAME NAME : FIRST

TEXT TO BE ENTERED (X,Y) POSITI

GRAPHIC OBJECTS LIBRARY USED POSITION (X,Y)

TO ENTER MORE TEXT OR GRAPHIC OBJECTS PRESS →

SBNO 2 LESSON NAME EXAMPLE FRAME TYPE . QUESTION & ANSW
FRAME NAME SECOND QUESTION TYPE · One Word

ENTER TEXT TO BE DISPLAYED: TEXT SIZE

GRAPHIC OBJECTS LIBRARY USED POSITION (X,Y)

FEED BACK ·

TO ENTER MORE TEXT OR GRAPHIC OBJECTS PRESS →

Figure 3: Storyboard part 2

system and can be used as front end tool for any system. The ideal situation in a large production environment would be to prepare the necessary graphic images by the graphic designers as done by AIP, and enter the sequence of lesson details via part 2 of the storyboard form, shown in Figure 3. This type of approach will provide excellent documentation in the development cycle and smooth running of the whole project. As well said by Palmondon and Deschenes (1986), consistent documentation cannot be kept during the development cycle of a courseware without automation of the most mundane task of editing.

Lesson Preparation

In the case of WISE lessons, keystroke files for the lessons could be generated by the preprocessors using the data entered in the storyboard forms. Using the input stream redirection facilities (WICAT, 1983) of WMCS, these keystroke files are input to WISE to generate the lessons automatically. Similar preprocessors can be written for the authoring systems like AUTHOR and FSL , an authoring language under development (Quaine & Johnson, 1986), and data entered in the storyboard forms can provide the variable input to these authoring languages. It could be investigated whether the keystrokes for creating each frame or event type could be captured using the operating system facilities of other authoring systems. If it is viable, then preprocessors can be developed similar to the user interface tools for WISE. Whether the preprocessors are essential or not depends on the users of a particular authoring system. Even if the preprocessors are not available it will be easy to transfer instructional material from well documented storyboards

Discussion

Most of the authoring systems aim at preparing similar type of instructional materials. Thus providing the automation of storyboards as an authoring aid to these systems can standardise the production of CBI material. This need not be considered as a front end tool only for the authoring systems. This can also be used as a front tool for developing audiovisual instructional materials similar to the IBM's Storyboard software package (Robbins, 1986). Our survey of different authoring systems does indicate that storyboards can be generalised as a distinct data type to be processed electronically on a wider environment, and so provide a general tool for effective instructional design. The development of such tools offers promise of efficiency in the authoring cycle, and further author support through electronically accessible high-level documentation. Such documentation can be ported to other sites of development and with slight modification to the material it could even be adapted to another system. As suggested by Bork (1984) this documentation can be sent to reviewers and experts which will enable the material to be reviewed before it is collated into a courseware using an authoring system. With most of the authoring systems it is tedious to modify the linkage between the frames once a large number frames have been produed. In the storyboard board processing the links set up can be easily modified and reviewed thoroughly before it becomes CBI courseware. This labour intensive process of instructional design and development can be made more cost effective by further development of Automated Instructional Systems similar to the Courseware Instructional Toolkit developed by Courseware Inc Kearsley, 1986. (This is a prototype system and the current version runs on IBM PC/XT.) A number of organisations are moving towards using CBI for training purposes. These types of automated support tools can provide a consistent methodology , re-usable design and cost-effective computer based teaching programs in a training environment (Howes & Williams, 1986).

Acknowledgements

We acknowledge with gratitude the assistance given to S. Sampath by the members of the CBT branch of AIP (Melbourne), Expert Solutions, Australia (Sydney), Telecom (Melbourne) and TAFE College (RMIT). All offered useful infomation and provided the opportunity to investigate different authoring systems and sample courseware.

Reference Note

1. Expert Solutions Australia, Private Communication.

References

Anver, A., Smith, S., & Tenczar, P. (1984). CBI Authoring Tools: Effects on productivity and quality, Journal of Computer-Based Instruction, 11(3), 85-89.

Bork, A. (1984). Producing Computer Based Learning material at the Education Technology Center, *Journal of Computer-Based Instruction*, 11(3), 78-81.

Data Training (1986). Annual survey of CBT authoring systems, *Data Training*, May, 32-68.

Dear, B.L. (1986). Artificial Intelligence techniques: Applications for course development, *Educational Technology*, 26(7), 7-15.

Garton, R., Reed, M.J., Reed. G. & Stevens, E. (1984). Developing a CBI Course: The process, Proceedings of 25th International ADCIS conference, pp142-143.

Hillelsohn, M.J. (1984). Bench marking authoring systems, Journal *of Computer-Based Instruction, 11*(3), 95-97.

Hofstetter, F.T. (1985). Prespectives on a decade of Computer-Based Instruction: 1974 84, Journal *of Computer-Based Instruction*, 12(1), 1-7.

Howes, R.F. & Williams, D.O. (1986). A methodology for developing Computer-Based teaching programs, Computing *Education*, *10*(3), 347-352.

Kearsley, G. (1986). The Courseware Instructional Toolkit™: A Prototype Automated ISD System for Personal Computers, Proceedings of 26th International ADCIS Conference, pp199 -204.

Lommel, J. (1986). What matters in an authoring system?, Data Training, July, p39-46.

Macleod, C. (1983). The Design and Development of Computer-Based Training Programs, *Training and development in Australia*, *10*(2), 11-16.

Merrill, M.D. (1987). Prescriptions for an authoring system, Journal of Computer-Based Instruction, 14(1), 1-10.

Merrill, M.D. (1985). Where is authoring in authoring systems, *Journal of Computer-Based Instruction*, 12(2), 90-96.

Plamondon, R. & Deschenes, J. (1986). Course design using software engineering methods, *Computing Education*, 10(4), 417-427.

Quaine, A.J. & Johnson, C.W. (1986). Frame sequencing control in PAL, Proceedings of CALITE 86, pp234-244.

Quaine, A.J. & Sampath S. (1986). User Interface Tools for WISE, Proceedings of CALITE 86, pp245-253.

Robbins, G. (1986). Cal redefined : a window on available resources, Proceedings of CALITE 86, pp13-22.

Romiszowski, A.J. (1986). Developing Auto-instructional *Materials*. London. Kogan Page, 54-94.

WICAT Systems, Inc. (1983). WMCS, Systems Programmer Reference Manual, Vol. 3.

S. Sampath is a Lecturer in the Australian Defence Force Academy with a research interest in Computer Aided Instruction and who teaches Information Systems and Programming Languages.

A.J. Quaine is a Senior Lecturer in the Australian Defence Force Academy with a research interest in Computer Aided Instruction and who teaches Operating Systems and Programming Languages.

Computer Managed Learning — Its application to increase student achievement using formative self-assessment

Jon D. Stanford, University of Queensland
Howard P. Cook, Queensland Institute of Technology.

Computer based assessment was introduced into an introductory economics subject, EC110, in the Department of Economics, University of Queensland, during Second Semester 1986. Over 600 students used computer generated and marked random tests for a number of assignments.

A commercial Computer Managed Learning (CML) software package was used to issue and mark student tests and to keep records, generate reports and analysis of student and question bank performance. Because tests were answered by students away from the computer, only a small number of terminals were required The distinction is made between Computer Managed Learning and Computer Assisted Learning (CAI).

The effectiveness of the CML system is evaluated by using evidence from the responses to a Student Questionnaire and by the end-of-year examination.

The paper concludes that CML provides a powerful mechanism for addressing such issues as learning effectiveness, u zation of resources, student motivation, support to students, learner control and cost-effectiveness.

It is recommended that tertiary institutions give serious consideration to the implementation of CML on a wide scale in order to improve student performance and staff productivity. This can be done at affordable cost.

This paper describes the implementation of CML in a large first-year economics course at the University of Queensland in Second Semester 1986. Although the introduction of CML arose from a particular set of circumstances and although CML was initially thought of as a trial run for 1987, CML was very effective and even in its first-up format was successful in promoting student learning. The introduction of CML in 1986 led to much more widespread adoption in the Department of Economics in 1987.

The particular circumstances occurred because one of the authors, Jon Stanford, had returned to lecturing the first- year course in 1985 and was interested in improving student performance and student motivation; as well he had a strong background in instructional design, use of computer simulations, and use of various media and course evaluation at the University level. The other author, Howard Cook, on secondment to the Computer Assisted Learning (CAL) Unit, University of Queensland, to advise university staff on the application of CML had a strong background in applications of CML for industrial training. Co-incidently the Faculty of Commerce and Economics, University of Queensland, had just installed a VAX computer which was available for use. The CAL Unit had purchased a commercial CML software package (CBTS) which operates on the VAX range of computers. Thus all the ingredients required for CML implementation, hardware, software, systems support, instructional design and over 600 students were available. The implementation worked well because the authors were able to work co-operatively and to enlist support from many sources.

The problems of organising, teaching and managing a large first-year course are substantial. Most students in the course enter university direct from school and many find the transition difficult. Many students find sitting in large lecture theatres alienating and in such an impersonal environment lose motivation. Academics are generally swamped by day-to-day concerns of administration of a large course: making sure student handouts are ready on time, dealing with student enquiries, liaising with tutors, setting tests and examination papers, maintaining records of student performance and progress; they are unable to take a detached look at the course and what they should be doing, and are unable to plan rationally for course development and for course innovation. Australian universities have been under financial pressure for some years and are unable to devote increased resources to overcome these problems.

The solution to some of these problems lies in the adoption of a new approach which will encourage students to take greater responsibility for their own learning, to set and achieve their own learning goals and to acquire good learning skills which will stand them in good stead during their professional lives. Consequently the implementation of a CML system within a large course is more than the use of a computer in teaching; it necessarily embodies the adoption of a different approach to learning and teaching in which the role or the teacher is that of facilitator of learning and manager of learning experiences. As Jon Stanford often remarks "the first thing to learn about computer managed learning is that it has very little to do with computers."

Definition of CML

Computer Managed Learning is defined here as the application where the computer does not have an instructional role and where the function of the computer is in the control, administration and testing of the learning process. Management of learning is the process of course design, specification of learning objectives, issue and marking of tests, recording results of tests, provision of feedback and remedial support, analysis of module effectiveness, analysis of learning performance, preparation of reports to students and lecturers (Miller 1986).

There is an important distinction between Computer Managed Learning and Computer Assisted Instruction. Computer Assisted Instruction is taken as the learning process in which instructional material is stored in the computer and released to a student seated at a computer terminal. Computer Managed Learning is used to refer to the system where instructional material is available only externally to the computer and where the computer plays an important management function. It is possible for a CML system to incorporate Computer Assisted Instruction but it is generally not necessary or even desirable.

The Course

The course, EC110 Introductory Economics, is a standard principles course which accounts for 25 per cent of the workload of a full-time student. The course is designed for continuing students in economics; the Department offers as well a terminal first year course for students who do not intend to proceed to the study of economics at second-year level.

The course, EC110 Introductory Economics, is required for all students enrolled in the Facu ty of Commerce and Economics (unlike most other Australian universities both the B Com and B Econ degrees are offered by the University of Queensland) and is available to students in other Faculties, especially Arts, who wish to enrol for a non-terminal course in economics. Total nominal enrolment in 1986 was 640.

The major features of the course enrolment can be summarised as follows:

- approximately 50 per cent of students are enrolled for the B Com degree;
- approximately 25 per cent of students are enrolled for the B Econ degree;
- approximately 25 per cent of students are enrolled for the B A degree or another degree;
- he entry requirements for the B Com degree are significantly more stringent than for the other degrees;
- approximately 75 per cent of students have entered direct from Queensland Grade 12;
- a small but significant proportion of students have experience of successful tertiary study.

The course is taught by large lecture format to three self-selected groups, two of which are scheduled during the day and the other scheduled in a block in the evening. The course is taught throughout the year i.e. over the two semesters, and is examined by two formal unseen examinations, each of two hours duration, in November.

The lectures are given by three academics; one of whom teaches the First Semester and the other two each teach half of Second Semester. The end-of-year examination is divided into three parts to reflect this division of lecturing effort.

The final grade for the course is determined by a combination of progressive assessment and end-of-year examination with a relatively high weighting towards the end-of-year examination. Although the determination of the final grade is relatively complex it can be conveniently explained in Table 1.

Table 1. Contribution of Each Form of Assessment to the Final Grade in EC110 in 1986.

Form of Assessment	Number of Marks
First Semester:	
Best Five Class Quiz Marks	25
Two Assignments	10
End-of-Year Examination	65
Second Semester:	
Best Five Class Quiz Marks (Part A & Part B)	25
Assignment - Part A	5
Assignment - CML Test	5
End-of-Year Examination - Part A	30
End-of-Year Examination - Part B	35*

* This mark was scaled according to the student performance in the final two CML tests.

The procedure is illustrated in Table 2 on the assumption that the maximum marks on the CML tests had been obtained:

Table 2: Illustration of Scaling of End-of-Year Examination Marks, Part B, EC110, 1986.

Unadjusted Mark		Scaled Mark	CML	Scaled Mark	
Out of 35 Marks	%	out of 25 Mark		out of 35	
35	100	25	10	35	100
30	86	21	10	31	89
26	75	19	10	29	83
25	71	18	10	28	80
23	65	16	10	26	74
20	57	14	10	24	69
18	50	16	10	23	66
15	43	11	10	21	60
10	29	7	10	17	49

The first CML test was in lieu of an assignment; the other two were voluntary; marks were obtained on an "all or nothing" basis. During 1986 there were no small group tutorials available in the course; students were required to complete a weekly class quiz and the results of this quiz were discussed in a "mass tutorial" following a lecture.

Learning Objectives for the Course

Previous experience with students in EC110 (and with later year economics students) has indicated that, in general, students could perform at high levels in tests involving verbal recall but that in tests requiring the display of other skills they performed at significantly lower levels.

Areas where students have performed at low levels are:

a. identifying the empirical counterpart of concepts understood in verbal terms;
b. drawing logical inferences from verbal statements or from simple data (either hypothetical or real);
c. drawing strong conclusions from verbal statements or from empirical data;
d. performing relatively simple calculations;
e. developing detailed critical analysis of theoretical or empirical topics.

Testing through the CML system was designed to improve student performance in areas a, b, c and d above and it was recognised that other responses were required to improve student performance in area e; in this area students have to acquire skills which involve analysis, selection, discrimination, synthesis, criticism and evaluation.

This implies the use of a two tier approach to learning in which the competence principle was adopted for the first tier where basic level skills are acquired. This approach was seen as appropriate in the course, EC110 Introductory Economics, where competence in basic level skills comprises the great majority of student effort. The competence approach required the division of course material into relatively self-contained modules and the providing students with clear cut learning objectives and specific instructions as to how to achieve those objectives. An inability to achieve these objectives at high levels (i.e competence) at the first attempt was not regarded as "failure"; instead learners are "recycled" until they attain competence. This implies that learners may proceed at their own pace in achieving learning objectives although institutional and logistical constraints place limitations on self-pacing.

The achievement of learning objectives at the first tier provide the ground work for acquisition of skills at the second tier level but the tasks set for learners to acquire second tier skills should not involve repetition of tasks used in mastery of first tier skills but should be designed more explicitly to achieve the particular learning objectives of the second tier.

This design of the course uses principles of learning derived from the Keller Plan approach to learning and from the general principles of a Personalised System of Instruction.

Contribution of the CML System to Achieving Learning Objectives

A CML system is able to make a significant contribution to assisting students to achieve the learning objectives of the course in the following ways, by:

generating unique tests for each student
marking the tests immediately
allowing the lecturer to monitor individual student
progress through the course
recording student marks and other data
providing various reports
providing detailed statistical analysis of student performance
generating detailed statistical analysis of the answers to items in the test bank so that testing procedures can be modified in the light of experience and of revealed difficulty.

A CML system offers significant advantages in that computers have a distinct advantage over people in doing repetitive tasks, in recording data and in undertaking large calculations and in collating information.

The testbank consisted of the following style of questions:

true/false
multiple choice

matching
completion
calculations.

Some typical questions and their relationship to the learning objectives are given:

Matching Question:

"Consider the following statements:

1. Exports of coal from Australia to Japan;
2. Import of an IBM computer to Australia;
3. Payment of interest on an overseas loan owed by the private sector in Australia;
4. Payment of an age pension to an Australian citizen living in Greece;
5. Purchase of an airline ticket by a resident of Hong Kong from Qantas for travel from Hong Kong to Singapore;
6. Purchase of oil by a Liberian registered ship in an Australian port;

which category would these be classified as, namely (G) goods, (S) services, (I) income, (T) transfers. (Enter your answer as either G, S, I or T. Note: Some categories may not appear, others may appear more than once.)"

This question is related to learning objective a; i.e. "identifying the empirical counterpart of concepts understood in verbal terms."

True/False Question:

"Given the following hypothetical data on exchange rates in respect of the Australian foreign exchange market:

	Yen/$A	Brazilian Cruzeiro/$A
June 1985	310	3560
June 1986	270	4920

The Japanese yen has depreciated against the $A?"

This question relates to learning objective b. i.e. "drawing logical inferences from verbal statements or from simple data (either hypothetical or real)."

Calculation Question:

"Assume you are a portfolio manager with $A ### to place in the short term money market and that the spot Dutch Guilder/$A rate is #.## and that money market rates in Amsterdam and Brisbane are ##% and ##% p.a. respectively for a six month term; what six month forward exchange rate would leave you indifferent between placing funds in Amsterdam or Brisbane? (Express your answer to two decimal places e.g. X.XX)"

[Note: The numbers represented by ### are randomised variables so that each calculation is unique.]

This question illustrates the proposition known in economics as " the covered interest parity theory of foreign exchange" and is important for understanding the relationship between money markets and foreign exchange markets. The calculation is fundamental to understanding the concepts involved.

Implementation of the Project

The project used a VAX computer and the software supplied commercially by CBTS with a testbank specially written by Jon Stanford. The course material was divided into three modules:

- International Trade and the Balance of Payments
- Foreign Exchange

- The Monetary System and Monetary Policy.

A further subdivision of the modules may have been desirable but the limitations of the time available for the project (the final seven weeks of Second Semester) and the large number of students made this impracticable.

Students were required to take three randomly generated tests; one for each module. Each test consisted of approximately 15 questions being a mixture of true/false; multiple choice; matching and calculation with each type of question carrying a different weighting. The mastery level was set as 75% for the first test, 80% for the second test and 85% for the third test. Students were allowed a maximum of three attempts at each test; each test had different randomly drawn questions.

Student names and identifiers (the first six digits of their student number) were loaded onto the VAX from student master files held on the University's IBM mainframe avoiding the need to enter such data manually.

On a terminal students were able to:

- draw a test
- have test marked
- examine the learning objectives
- set/change a password
- send a message to the lecturer
- look at their records and status.

If students drew a test the computer generated one which was then queued for printing at one of the two teletype printers and logged the student off the system. If students took one of the other options they remained on the system for longer but for a relatively short period.

The system had 14 VDU terminals and this proved to be generally satisfactory for the 600 students who used the system; there were queues at peak times especially lunch time but these were no longer, and nearly always shorter, than those in the student Refectory. The Computer Room was open 8am-10pm, Monday to Friday and 9am-5pm Saturday and Sunday.

Students prepared answers for the test away from the computer; some answers required consultation with the text books, lecture notes, handouts and required calculations to be performed. When ready to answer the test students logged on to the system and gave their answers when prompted by the computer which provided opportunities to review and change answers. When the answering process was completed the computer marked the test, informed the student of the percentage achieved and printed out the correct answers to incorrectly answered questions. If students obtained the required percentage they were able to draw the next test; if not, they could draw another test on the same module.

The lecturer was able to monitor student progress from reports supplied on demand by the CML system, to respond to individual messages from students and to give messages to the entire class.

Discussion of Student Questionnaire

Details of the Student Questionnaire are available in a detailed report of the project; some issues raised by students which require further elaboration are discussed here. These are:

- system problems
- test design and construction problems
- cheating
- formative versus summative testing.

The second surprise from the project (the first being the extremely high level of student support for CML) was the relative small number of system problems. Most related to printer and paper supply problems; given the demands on the printers (over 3500 tests and 3500 results were printed in six weeks) and the poor quality printers used, the problems were well within acceptable bounds however annoying and frustrating they were to the students concerned. The solution to this problem is simple - buy better quality printers! (This has been done at low cost in 1987 and this area of difficulty has been significantly reduced).

The level of problems with the test design and construction was also low; some problems were easily fixed - errors in questions and solutions were eliminated as soon as they became known and some test items were refined in the light of immediate experience; however, the temptation to tinker with the testbank was resisted until the CML testbank analysis had sufficient data.

Some students commented that questions were "ambiguous"; this may have been true of some questions but most of the "ambiguous" questions showed that some students were unable to discriminate between finer shades of meaning, closely related concepts and valid and invalid inferences.

A small number of comments related to the possibility of "cheating" because the tests were done away from the terminal without supervision. Since the testing was designed as formative rather than as summative, students were encouraged (at times required) to consult textbooks, notes and references to answer test questions correctly. As well, some test items generated spirited (at times heated) debate in small groups of students. We think that such activity is entirely appropriate at a university and consider that it is a desirable feature of the CML system if it encouraged such behaviour.

It is possible under the testing procedures for a student to cheat in the ordinary sense of the term i.e. to obtain answers from another student and use them for their own. However, the system has safeguards against this.

First, each test is unique (calculation questions have been randomised so that even if two students receive the same question the numbers are different) so that collaboration is made difficult. This is particularly true in the first year of CML when all the questions are new and there is a high level of uncertainty about the correct answer.

Secondly, while there are advantages to cooperation between students, it remains true to say that students are in fact competing for grades so there is an incentive for better students to withhold information.

Thirdly, to cheat in this way requires the cheater to be able to identify other students with superior and correct information and who are willing to release this knowledge; empirically we can say that such abilities to discriminate are rare.

Fourthly, and perhaps more importantly, it should be understood that the CML system and associated computer held records provide an easy means for the lecturer to maintain surveillance of student performance and to identify students whose performance on the CML tests appears to be inconsistent with other test results and student characteristics.

Fifthly, assessment of the end-of-year results suggest that very few students passed the course by cheating in the CML tests. This issue is being examining closely in the study of student performance now being undertaken.

The student questionnaire

QUESTION 1: Before using the CML system did you have any previous experience with computers?

QUESTION 2: What did you think about computer based (immediately available) testing?

QUESTION 3: What did you think was the greatest advantage of the CML system?

QUESTION 4: Did you experience any frustrations with the CML system ?

QUESTION 5: Would you have liked the CML system in the entire EC110 course?

QUESTION 6: Would you like to see the CML system in other courses?

QUESTION 7: Do you have any other comments about your experience with the CML system?

QUESTION 8: What is your verdict on the CML system? Should the Department proceed with this system?

QUESTION 9: Would you like to have more control over the CML system (e.g. drawing tests whenever you liked; proceeding through the course as fast as you liked)?

QUESTION 10: Would you please tell me something about yourself?

QUESTION 11: Any other comments?

Evaluation of CML

The evaluation of CML was carried out by assessment of the responses to the Student Questionnaire, the results of the End-of-Year Examination and the cost of CML.

The responses to the Student Questionnaire

The response to the Student Questionnaire was over 75 per cent of the nominal student enrolment and comprised the following by degree enrolment:

	Per Cent
B Econ (incl. B Econ/LLB)	20
B Com (incl. B Com/LLB)	53
B A and other degrees	28

This is representative of the total enrolment.

In answer to Question 5: "Would you like to see the CML system in the entire EC110 course?" 91 per cent of respondents said yes; in answer to Question 6: "Would you like to see the CML system in other courses?" 87 per cent of respondents said yes; while in answer to Question 8: "What is your verdict on the CML system? Should the Department proceed with this system?" 96 per cent of respondents said yes.

From this evidence we conclude that the student satisfaction with the CML system was at a high level and that student response to the system was positive.

From the general discussion of the responses to the Student Questionnaire we conclude that generally the CML system constituted a positive learning experience for most students.

The results of the End-of-Year Examination

The unadjusted and unscaled results f the unseen end-of- year examinations a shown in Table 3 below; the examination papers were set by each academic responsible for lecturing each segment of the course but were not necessarily marked by that person, in fact most of the papers were marked by other members of the Department.

Grades in the University of Queensland are awarded on a seven point numerical scale from one through seven with a grade of four being a Pass and grades of five, six and seven being superior passes in ascending order. A grade of three (usually awarded to papers with a mark just below 50 per cent) is a non-qualifying pass while grades of two and one are unambiguously failures. The examining conventions at the University of Queensland as expressed in the guidelines issued by the then Professorial Board (now the Academic Board) are that grades should approximate the percentages indicated in Table 3. Final grades for the course are determined by results of progressive assessment as well as end-of-year examinations.

Table 3. Results End-of-Year Examination in EC110 Introductory Economics, 1986.

	Percentage of Students			
Percentage Mark	First Paper Grade		Second Paper Part A	Second Paper Part B*
0-40	1	11.5	47.6	14.8
41-49	2/3	20.4	10.1	4.2
50-64	4	43.3	21.3	19.3
65-74	5	17.6	10.1	18.5
75-84	6	6.7	8.5	24.4
85+	7	0.7	2.4	18.8

* Segment of the course to which the CML system applied.

In order to assess the results in Table 3 it is necessary to consider the 'a priori' expectations one would hold in respect of the three components of the end-of-year examination. Since the segment of the course to which the CML system applied and to which the Second Paper Part B results apply was the last segment of the course and because some knowledge of the segment covered by Part A is considered necessary as a pre-requisite for Part B, our 'a priori' expectations are that results in Part B would be no better than in Part A or in the First Paper and may be significantly worse.

In the light of this expectation, our interpretation of the results in Part B, which are statistically significantly different from those in Part A and the First Paper, is as follows:

1. the results in Part B are consistent with previous experience in Keller Plan and PSI courses;

2. the results in Part B discriminate clearly between students with passing and failing grades which we regard as a desirable feature in its own right;

3. in 1986 this discrimination meant that very few students would have improved their adjusted mark in Part B as a result of the CML tests as can be seen by consulting Table 2;

4. the results of Part B may reflect a Hawthorne effect due to strong student acceptance of CML as disclosed in the results of the Student Questionnaire so that the superior results in Part B may be due in part to inducing students to devote more time and effort to the work of Part B.

5. given these qualifications and other matters which cannot be tested, we would conclude that the results in Part B are consistent with the view that student performance in Part B of the course has been improved through implementation of CML.

The Cost of CML

The costs of CML fall into the following categories: hardware costs, software costs, costs of establishing the testbank and the operational procedures and material running costs.

Hardware costs

Under the University of Queensland's resource allocation and accounting procedures, the costs, both initial capital and running, are not allocated to operational units' budget and the hardware is treated as a common resource so it is not possible to know how much of the hardware cost is properly allocable to this CML project. For the purposes of making cost comparisons we assume that the rental costs of the VAX for the period of the project were $25,000.

Software costs

The all-up cost of the CML package, supplied by CBTS in 1985, was $18,000.

Costs of establishing the testbank and the operational procedures.

Major costs of the project were the cost of establishing the testbank, which Jon Stanford estimates as taking at least three weeks full-time work, and the costs of setting up the project which took two months of Howard Cook's time. Ensuring the project worked meant that both authors had to forget any ideas of 9 to 5 days, and during the period CML was used by students, both worked weekends and most nights of the week.

Material running costs

Costs for consumable supplies to keep the project running (paper and ribbons) were trivial. Total costs were less than $500 and amounted to an average cost per test generated of approximately 11 cents. This cost also included the processes of marking the test, issuing results to the student, keeping records and providing data for analysis of student and question bank performance.

Evaluation.

Our evaluation of CML suggests that it is cost-effective; if we allocate all costs, except for the time of the authors in developing the project, to the first year of the project. we estimate the average cost per student is approximately $70. If the cost of the software is amortised over three years, the average cost falls to $50; if the cost of the hardware is amortised over the expected life of the hardware, the average cost falls to approximately $30. These costs have to be compared with the marginal funding available to the Department in respect of each additional student; in 1985 this was $2500 per student or over $600 if allocated by the straight line method to EC110.

That CML is cost-effective is emphasised when it is remembered that the establishment costs (first year costs) are high, and second and subsequent year costs are much less than the initial year costs. In addition, in later years, CML can be routinised so that it can be maintained and expanded with less skilled and dedicated staff than is required to establish it in the first place. In 1987 CML is being used in three first-year courses and one third-year course; the costs of expansion in this way are much less than the establishment costs.

Other issues

CML enables more students to be taught effectively with given resources; it enables a saving of resources used in the repetitive areas of marking simple tests and in recording results and other data; these resources can be more effectively used in other areas of course management and design. CML also enable students to gain more control over the pace at which they progress through the course; if this feature of CML were expanded we estimate that a significant proportion of students could complete the course requirements in less than a semester.

In order for these benefits of CML to be captured completely, the University needs to change certain features of its organisation. One horror story will suffice to illustrate this point. The complete student records in EC110 were maintained on computer in 1986, the University administration was offered the lists in any one or all of the following ways:

- in hardcopy printed by computer in any format;

- on floppy disc;
- on file on the University mainframe transferred via Kermit from our floppy or from the VAX.

We were amazed to find that the only format they would accept was their forms completed by hand!

Conclusions

From the experience of this project of using CML for formative assessment in a large first-year economics course in 1986, we conclude that:

1. It had improved student performance by as much as 40 per cent on the final examination in 1986 (we can also note that it significantly reduced the percentage of students who obtained less than 50 per cent in the examination as compared with the results of in 1985);

2. Students had positive responses to CML;

3. Students found that CML supported their learning; however, many requested increased learner control from CML;

4. Student motivation in the course increased;

5. CML was cost-effective;

6. The costs of establishing CML are relatively high; but CML offers significant cost advantages in later years of use;

7. A major cost of establishing CMI is the use of skilled and dedicated staff time;

8. CML can improve productivity in teaching a large first year course in two ways: through allowing academic staff to manage more students with given resources and in enabling students to progress through the course more quickly and at a higher level of achievement;

9. In order to reap these benefits organisational changes will be necessary.

Acknowledgements

The authors would like to acknowledge the assistance of Dr B.W. Carss, Dean, Faculty of Education and Co-ordinator, CAL Unit, University of Queensland, Mr C.W. Farmer, Manager Employee Relations, QEC, and Mr F.J. Miller (then with QEC) in arranging the secondment of Mr H.P. Cook to the CAL Unit. The Head of the Department of Economics, University of Queensland, Dr A G Kenwood, was very supportive of the project.

The particular assistance of Mr M.J. Wood, Faculty Programmer, Faculty of Commerce and Economics, University of Queensland, whose contribution in setting-up and maintaining the CML hardware and software had significant bearing on the success of project.

Reference

Miller F.J., Cook H.P. and Clark C.Q. (1986). The Use of Computers in the Maintenance of Workforce Competence, Electric Energy Conference 1986, Institution of Engineers Australia, Brisbane, 257-258.

Jon D. Stanford, Senior Lecturer, Department of Economics, University of Queensland. Major Research interests are Monetary Economics, Economic Policy, Public Sector Economics and Economics Education. In the area of Economics Education previous projects include use of the media in teaching economics, course and instructional design, course design and evalu-

ation in distance education. Current projects include a study of student performance in first year economics and the application of CML in teaching Monetary Economics at third year level. Previous work in Economics Education has been published in Teaching at a Distance, The British Journal of Educational Technology, ASPESA Newsletter and in O. Zuber-Skerritt (ed), Video in Higher Education, London, Kogan Page, 1983.

Howard P. Cook, Manager-Projects, Queensland Centre for Computer Managed Learning (Q-CML), Queensland Institute of Technology. Research interests include the use of CML to manage the learning process and the application of resource based learning packages to accommodate individual learning needs particularly within the industrial/commercial environment. Previous CML experience includes co-researcher of a National Training Council project in CML in a large decentralised industrial organisation. Current activities include the development of a commercial CML system operating on PCs.

Computers in tertiary education in the northern territory — A review

Jean Stevens
Department of Computer Science
Darwin Institute of Technology

There is a general lack of knowledge in the other States of Australia about current developments in the Northern Territory. Those who visit the Northern Territory are surprised to find the pervasiveness of computing throughout all levels of education. Computing technology has been embraced by a population characterised by isolation from the rest of Australia.

This paper reviews the facilities, availability and use of computers in tertiary education both on campus at Darwin Institute of Technology, the University College of the Northern Territory, and the Alice Springs College of Technical and Further Education, and off campus through the Northern Territory Open College and the Northern Territory External Studies Centre.

The Northern Territory has a population of 160 000 people who are isolated from the rest of Australia. In spite of its small population and the huge area it covers, computers have become widespread throughout all levels of education. The Northern Territory has five organisations which provide Advanced Education, Technical and Further Education and facilities for external studies in computing. Government departments are closely associated with these activities, and in spite of the difficulties imposed by isolation, there is a healthy interest in research and post-graduate studies. and Computing in the Northern Territory is very much alive and well.

Darwin Institute of Technology (DIT), originally established as the Darwin Community College in 1974, provides the central focus for a large proportion of the many computing activities in the Northern Territory. The primary teaching facilities of the Institute are provided by a VAX Cluster and, in addition, many exciting developments have taken place with a number of joint educational facilities being set up between DIT, government departments and other organisations. These include the establishment of an Information Technology Training Centre, a Computer Education Centre and a research and development group, the Special Interest Group - Computer Oriented Nurses (SIGCON), specifically concerned with the relationship between computers and the nursing profession.

Information Training Technology Centre (ITTC)

The Information Technology Training Centre (ITTC) is a joint facility provided by Darwin Institute of Technology and the Northern Territory Treasury, Division of Computing and Information Technology (NCOM) which is responsible for providing suitable computer training courses for all Northern Territory Government employees. The Centre provides up-to-date IBM equipment for use by Institute lecturers and students and training in microcomputer and mainframe applications for Public Sector employees.

The centre is situated in the Faculty of Business and consists of a Mainframe Terminal Laboratory and a Personal Computer Laboratory. The Personal Computer

Laboratory is equipped with 10 XTs, each with single disk drives and a hard disk, and the Mainframe Laboratory has 12 telex terminals connected to the Northern Territory Government IBM 3090 computer. All of the courses offered in the Centre concentrate on giving "hands on" experience with the relevant system or package.

The courses utilising the 3090 are mainly for Public Sector employees, who use the Government networking facilities. The courses offered cover a diverse range of materials including EASYTRIEVE+ and SPSSX. Courses utilising the Personal Computer Laboratory cover material such as SYMPHONY and SMART and also a SMARTLINK course, which is designed for mainframe to microcomputer communications in the GAS system. These courses are usually one or two days in duration and form a sequence which are taught by either DIT lecturing staff or by NCOM staff. DIT computing students have access to the ITTC laboratories where appropriate.

The Computer Education Centre

The Computer Education Centre is a joint facility provided by Darwin Institute of Technology and the Northern Territory Department of Education. The Centre coordinates the use of computers in education in all Northern Territory schools and is responsible for providing the following services to the Education Department, D.I.T and the community:

- inservice training in computer use and curriculum applications;
- pre-service training in computer use and curriculum applications;
- post-graduate studies in educational computing;
- training for all members of the school community, including parents, in the educational uses of computers;
- software evaluation;
- maintenance of a software lending library in conjunction with the Education Department's Library Services;
- evaluation of new technology and machines to determine their applicability in schools;
- advisory visits to schools and regions to give curriculum and systems advice and inservice training;
- coordination and support for regional education centres;
- information dissemination through publications;
- liaison with national project teams for development of the uses of computer technology as an educational resource;
- resources for award and non-award computing courses offered by D.I.T. and
- fund-raising courses.

The Computer Education Centre operates as a functional unit of the Division of Computing and Mathematics and is situated in the Faculty of Applied Science. It consists of an Apple 11e Teaching Laboratory, an Apple MacIntosh Teaching Laboratory and a Student Access Room. In addition, the Centre houses a range of personal computers, such as the BBC Microcomputer and the Commodore 64 which are available in a number of Territory Schools, together with a number of other machines under review.

The Centre is currently staffed by two Senior Education Officers from the Department of Education, who are attached to D.I.T. as members of the academic staff, and one lecturer from the Division of Computing and Mathematics. Lecturers from the Department of Computer Science work closely with the Centre and utilise the facilities on a regular basis for teaching purposes.

SIGCON : The Special Interest Group - Computer Oriented Nurses.

The impetus for the formation of SIGCON came from a group of Registered Nurses who had completed a computing unit as part of a Diploma of Applied Science. The interest expressed in, and the lack of knowledge of, the computer as a tool for Health Care led to some initial research being undertaken in this area which has now expanded into the areas of knowledge-based systems and applications of intelligent CAL systems. Practising nurses, lecturers in Nurse Education and lecturers in Computer Science comprise the core membership of SIGCON.

Health care practitioners in the Northern Territory are faced with unique problems that require specialised training, particularly in areas such as Aboriginal Health and Community Health Care in remote areas. Members of the nursing profession regard the computer as a tool that could aid such training through CAL techniques and the use of knowledge-based systems and they feel that participation in the development of such tools will enhance their professional standing.

The group operates under the auspices of DIT and the Northern Territory Branch of the Australian Computer Society. It has access to the facilities of the Department of Nursing and the Computer Education Centre at DIT and is currently negotiating with Royal Darwin Hospital for research facilities. In addition to the research activities SIGCON publishes a regular newsletter and has run several workshops and seminars for its members.

Other Teaching Facilities in the Northern Territory

The Alice Springs College of Technical and Further Education (ASCOT) has three computer laboratories, one based on the Burroughs B26 minicomputer system and the other two based on personal computers. These are utilised by award courses offered in Business, Secretarial Studies, Management and Hospitality Management. ASCOT also runs seminars and workshops for outside users such as Government departments and fee paying courses. Modems are provided for access to other tertiary centres for external study purposes.

The Northern Territory External Studies Centre situated at DIT coordinates most of the interstate tertiary studies undertaken by Northern Territory residents, including the activities of a large number of computing students. The Centre provides tutors for many of these students and is currently co-ordinating the activities of a group of post-graduate computing students.

The newly established Northern Territory Open College is designed to provide post-secondary courses for isolated residents and uses the computing facilities of rural schools to provides a number of short courses in computer studies out of school hours. The Open College has two centres in the Darwin Rural Region and other centres in Katherine, Tennant Creek, Alice Springs and Nhulunbuy.

The University College of The Northern Territory (UCNT) has only been recently established. It is situated in Darwin and currently has a MicroVAX II and three teaching laboratories. At the moment these are used only for one semester introductory courses.

Although many advances have been made in computing throughout the Northern Territory, there is still potential for future development. It is anticipated that the School of the Air and The Northern Territory Secondary Correspondence School will provide facilities for remote users of computers. A link between Darwin Institute of Technology and the University College of the Northern Territory may be established to fully utilise and extend the facilities of both institutions and the Computer Education Centre intends to extend its services to the regional education centres through communication systems such as Viatel. The pervasiveness of computing throughout all levels of education will become even more evident in the future.

Appendix: Computing facilities at Darwin Institute of Technology

On 30th April 1987, Darwin Institute of Technology had approximately 1500 students enrolled in Advanced Education (degree or diploma level) courses and 3500 students enrolled in Technical and Further Education (TAFE) courses. In addition to students enrolled in award courses 3500 students attend fee paying courses at the Institute every year. Darwin has a population of 72000 so that the educational needs of nearly 12 percent of the population are met by the Institute.

Some 350 students enrolled in award courses have accounts on the main frame system and a similar number utilise the various micro-computer facilities.

Darwin Institute of Technology has a VAX/780, a VAX/750 and a micro-Vax which are clustered together and are utilised by staff and students.

The author has taught in a variety of institutions and countries and been Involved in computer education for over twenty years. The current position held by the author is that of Lecturer in Computer Science specialising in Technical Systems Analysis, Project Management and the Applications / User Interface. She is the convener and technical adviser to the Special Interest Group, Computer Oriented Nurses (SIGCON) which operates in the Northern Territory. This group is involved in computer education and research into the use and development of CAL systems related to the nursing profession. The author is currently developing an expert system which will enable nurses to gain advice on areas of nursing specifically relevant to the tropics. This system should be of great assistance to both nursing trainers and nursing staff in isolated areas. The knowledge required of nurses is much more general than that required of doctors, although not as in-depth. A major component in the development of the expert system is in discovering ways of extracting knowledge from people with no technical background in computing and in developing ways to access large amounts of general information in a reasonable time. Therefore the research falls firmly in the areas of knowledge engineering, knowledge representation and pattern recognition.

Computer assisted examination for fundamental programming courses

Tao Li, Department of Computer Science, The University of Adelaide
Luyuan Fang and Larry Christensen, Dept. of Computer Science, Brigham Young University, Chongsan Yu, Dept. of Electrical and Computer Engineering, University of Wollongong

In this paper, we present several types of questions which are suitable for computer assisted examination of fundamental programming courses. Computer implementation and analysis of the questions are also addressed here. Human factors in designing the examination questions are also considered. The choice of questions and the appropriateness of implementation are discussed. The analysis and argument presented here can justify our design and implementation.

It has always been a hassle for students and teachers at the end of each term or semester. Students are busy at preparing for examinations and the teachers are busy at making and grading examinations. Very often students have to come to a specified classroom at a scheduled time to take the exam and the teachers must be present at the exam. After the exam, teachers and graders must spend a large amount of time to design a grading scheme and to grade the exam papers. This has been quite a laborious and error prone job. Students often come to complain about their grades after the exam papers were graded. This raised our interests in implementing a computer-assisted examination system to deal with above problems.

The aims of computer-assisted examination are to give prompt processing of exam data for administrations and to reduce the labour and errors for grading exam papers. Computer-assisted examinations can provide flexible time and place for examination. In addition, properly designed exams can be error free and the grading can be completely computerized. Since we are discussing exams for computer programming courses, there are also certain questions which are natural to computer exam but not paper test.

However, not all examinations can be completely computerized in our current technology. Exam questions involving natural language descriptions are especially difficult to automate. Therefore, we must choose several types of questions which cover a large variety and are easily computerized. The simplest are the yes/no type of questions and multiple choice questions. These questions have been successfully used in TOEFL and GRE. But these questions are not sufficient for fundamental programming courses. Examinees may get a certain number of points simply by guessing and making random choices. These questions are also too simplistic to prevent cheating and plagiarism if the exam is to be taken by many people at different times.

In this paper, we consider several types of questions besides the yes/no type and the multiple choice type. We will also analyze the scoring probability of the questions in each type. The advantages and disadvantages of each type will be considered. Examples of each type will be given. Computer implementation of the exam questions will be discussed.

Types of Examination Questions

In this section, we describe several different types of exam questions. The scoring probability of each type is also analyzed. An example will be given for each type. For completeness, we also include the yes/no type and the multiple choice type. When we describe the types of exam questions, we also consider some 'human factors', which measure the effectiveness of the questions while being used against students.

YES/NO Type Questions

A question in this type requires a simple answer, either a YES or a NO. Even without an understanding of the problem, an examinee has a 50% chance of getting a correct answer. Following is an example question of this type.

> Question 1. In a digital computer, information is represented by a sequence of tones with continuous frequency increments?
>
> Answer: yes( ) / no( )

Analysis for yes/no questions is given below. From here on we will use 'character' for characteristics and 'prob and HF' for probability and human factor.

- (Character) This type of questions is simple and easy to grade. It is very useful to test student understanding of basic concepts.
- (Prob and HF) The 50% probability makes it susceptible to cheating and pure guessing.

Multiple Choice Questions

An answer to this type question usually requires the examinee to choose one from a set of n answers provided to him. Without an understanding of the problem, the examinee has 1/n chance of getting a correct answer. Since the examinees normally scan from the first answer tc the last answer, it would be advisable to place the correct answer for each question in the last few answers. Following is an example.

> Question 2. A recursive function is a subprogram which
>
> A. has multiple parameters
> B. can always be converted into iterative loops
> C. calls itself indefinitely
> D. can be implemented using stacks

Only one answer is correct in this case. With a random choice a student has 25% chance of getting the right answer. This is better than the yes/no questions. The analysis is as follows.

- (Character) Multiple choice questions are still simple and easy to grade. They are also useful for testing basic understanding. But this type of questions is trickier than yes/no type questions.
- (Prob and HF) A probability of 1/n is still susceptible to cheating and the answer to a specific question is easily memorised and passed onto another person.

Even though the yes/no type and the multiple choice type have certain disadvantages, they still prevail in exams. This is due to the simplicity of such types of questions.

Multiple Match Questions

This type of questions usually contains a description and two lists of items, each item in one list must be matched to an item in the other list. Sometimes we may arrange to have more than two lists for matching. Suppose that we have two lists, each of which has n items. If the examinee is totally ignorant about the question, the probability p(k,n) for him/her to get k correct answers from a total of n answers is approximately1/(ek!). This is because

$$p(k,n) = \frac{\text{total \# matches of k correct and (n - k) wrong}}{\text{total \# of matches}}$$

and the number of permutations with exactly k matches (PfeifferSchum, 1973) is

$$\binom{n}{k}(n-k)!\sum_{i=0}^{n-k}\frac{(-1)^i}{i!}$$

which is approximately n! e^{-1}.

$$p(k,n) = \frac{\binom{n}{k}(n-k)!\sum_{i=0}^{n-k}\frac{(-1)^i}{i!}}{n!} = \frac{\binom{n}{k}(n-k)!}{en!} = \frac{1}{ek!}$$

Apparently, the larger k is, the lower probability of getting it right. Following is an analysis for multiple match questions.

- (Character) Multiple match questions are simple to do and easy to grade. Partial grades are given according to the number of correct matches.

- (Prob and HF) Probability is 1/(ek!) The probability shows that scoring by random guess is very difficult. These questions are not susceptible to cheating and plagiarism because memorising so many matches is quite difficult.

Although multiple match questions appear similar to multiple choice questions, they have quite different human factors and probabilities.

Program Trace and Selective Types

One type of exercises, the trace of a program run, is particularly helpful for students in fundamental programming courses. The student must trace a program segment line by line to find out what are the output data. There are three typical types of trace questions, the direct output type, the selective choice type. and the selective match type. Each of these types will be examined next.

THE DIRECT OUTPUT TYPE. In this type of questions, a program segment is given. The student is supposed to trace the execution of the segment and to report the output sequence. Following is an example.

Question 3. Trace the execution of the procedure f(5). Show the output sequence produced by the write statement.

```
function f(n:integer):integer;
var t:integer;
begin
if n <= 1 then begin write(1); f :=
1 end
else begin t := n*f(n-1); write(n);
f := t end;
end;
```

The student is required to list the sequence of numbers output by the function with a parameter 5. A sequence of five numbers is produced by f(5) and the student must trace the program to produce this sequence. Analysis is shown below.

- (Character) This type of questions may range from easy to very hard depending on what are given in the program segments. Partial grades can be given according to the numbers of correct answers and wrong answers in the output sequence. Some questions can be very challenging.

- (Prob and HF) Probability for this type relies on many factors and is not easily calculated. But it is obvious that getting a correct sequence can be quite difficult. It is also hard to memorize the entire sequence and is thus less susceptible to cheating.

THE SELECTIVE CHOICE TYPE. In this type of questions, a program segment is given together with an output sequence. The student is supposed to trace the execution of the segment and to select the elements in the sequence which are produced by the segment. Following is an example.

Question 4. Which of the following numbers cannot be produced by the write statement in the following program segment.

```
i := 0; j := 1;
for k:=1 to n do  (* where n is greater
than 1 *)
begin
temp := i; i := j; j := i + temp;
write(i)
end;

Output                    numbers:
1,2,3,4,5,7,8,11,13,17,21,25,30,34
```

The student must identify, from the output sequence, the numbers which can be produced by the write statement in the program segment. The probability p(k,m,n) of getting k correct answers, from a sequence of n numbers among which m are correct answers, is

$$p(k,m,n) = \frac{\sum_{i=k}^{n} \binom{m}{k}\binom{n-m}{i-k}}{\sum_{i=k}^{n} \binom{n}{i}}$$

where $k \le m$ since there are a total of m correct answers only. The derivation of the probability is neglected to save space. Analysis is as follows:

- (Character) This type of questions also ranges from easy to very hard depending on what are given in the program segments. Partial grades can be given according the numbers of correct answers and wrong answers. Grading is easy.

- (Prob and HF)

$$p(k,m,n) = \frac{\sum_{i=k}^{n} \binom{m}{k}\binom{n-m}{i-k}}{\sum_{i=k}^{n} \binom{n}{i}}$$

It is bit difficult to comprehend this formula of probability. But it suffices to say that getting a completely correct sequence is very hard. Also this type of questions is not susceptible to cheating.

THE SELECTIVE MATCH TYPE. In this type of questions, several program segments are given together with several output sequences. The student is supposed to trace the execution of the segment and to select the elements in the sequence which are produced by the segment. An example is given below.

Question 5. Three programs shown below. A table is also given for filling in the answers. Each row corresponds to a specific value of i (the index variable) and each column in a row contains the output for one of the programs. Fill in the circles in each row with the corresponding program names (A, B, or C) to indicate the programs which produce the outputs.

```
program A;              program B;              program C;

var i,x,y,z:integer;    var i,x,y,z:integer;    var i,x,y,z:integer;
begin                   begin                   begin
for i:=1 to 5 do        for i:=1 to 5 do        for i:=1 to 5 do
 begin                  begin                   begin
 x:=round               x:=trunc                x:=trunc(sqrt(i /
 (sqr(i/2.0));          (sqr(i/2.0));              2.0));
 y:=x mod 3;            y:=x mod 3 - 1;         y:=x mod
                                                   3 + 1;
 z:=x*y;                z:=x*y;                 z:=x*y;
 end;                   end;                    end;
end.                    end.                    end.
```

index	x y z	x y z	x y z
i=1	0 -1 -1	0 1 1	0 0 0
i=2	1 2 3	1 1 2	1 0 1
i=3	2 1 3	2 2 4	1 2 3
i=4	4 0 4	4 1 5	1 2 3
i=5	6 -1 5	1 2 3	6 0 6

The student must identify, on each row, the program which produced the output in a column. Each row requires a match and the probability for getting k_i correct answers in the i-th row is $1/(ek_i!)$. Let k denote the total number of correct answer. Then $k \le n \times m$ where $k = k_1 + k_2 + \ldots + k_n$. Since the matching in each row is independent of the other rows, for a question with m programs and n rows of answers, the probability p(k,m,n) of getting k correct answers, is approximately

$$p(k,m,n) \doteq \frac{1}{\binom{n+k-1}{k}} \sum_{i=1}^{n} \frac{1}{ek!}$$

because the number of combinations for getting k correct answers out of n rows is equivalent to distribute k nondistinct object into n distinct cells [Liu,1968] and the result is $\binom{n+k-1}{k}$.

In calculating the above probability, we have assumed that the probability for each combination of k_i, where $\sum_{i=1}^{n} k_i = k$, is the same.

Following is an analysis of this type of questions.

- (Character) This type of questions are usually difficult to do. Partial grades can be given according to the numbers of correct answers. Grading is easy.
- (Prob and HF) p(k,m,n) is approximately

$$\frac{1}{\binom{n+k-1}{k}} \sum_{i=1}^{n} \frac{1}{ek!}.$$

This type of questions is not susceptible to cheating and is hard to do. But they are very useful exercises for learning to read and digest programs.

Our discussion on program trace questions stops here. Program trace questions are very important in fundamental programming courses. The types of trace questions we have chosen are easily computerized. We have implemented these questions and the results are satisfactory.

Debugging Questions

Debugging is an important step in programming. It is especially important for students in fundamental programming courses to learn about debugging. Debugging syntax errors are relatively easy and modern compilers normally report very detailed error messages. But it is more difficult to debug semantic errors. Hence, it is essential to include such questions in the examinations. An example of debugging question is given below.

Question 6. A recursive function FINDMIN is shown below. The function tries to find the minimum value element in an array X[1..N]. The value of the minimum element is returned. There are two (semantic) errors in the following program. Check the program and mark the erroneous statements. Also, give the correct replacements for the two statements.

```
type INARRAY = array[1..N] of integer;

function FINDMIN(var X:INARRAY;
N:integer):integer;
var min-of-rest: integer;                  |
 begin                                     |
                                           Erroneous
                                           Statements
                                               [#]
  if N=1 then FINDMIN := X[1]              [1]|
    else                                   [2]|
      begin                                [3]|
        min-of-rest := FINDMIN(X,N-1);     [4]|________
          if min-of-rest<X[N] then         [5]Corrected
                                           Statements
              FINDMIN := X[N]              [6]|
          else FINDMIN := min-of-rest;     [7]|
      end                                  [8]|
end;                                          |
```

This type of questions requires a close examination of the program segment and a careful check of each statement and each variable. The probability of locating the erroneous statements is $\binom{n}{m}$

where n is the total number of statements and m is the number of erroneous statements. It also requires much effort to correct the m statements.

Programming Questions

This type of questions require that the student to write a simple procedure or function and to run it on line. This also needs to provide a simple editor for on-line editing. However, there is no easy and fair way to grading. Either a simple no points/full points grading is employed or an off-line grading scheme is employed. We have also implemented this type of questions in our computer-aided examination system.

Implementation

The types of examination questions and some other types of questions have been implemented on a WICAT graphics workstation. To improve human-computer interaction and help the student during the exam, we have included many features to simplify the use of the system and to make it more user-friendly.

We have taken fully advantage of the graphics facility of the WICAT system. Along with each exam question, there are also various instructions for using the facility displayed. The user can use the system without referencing any manual or guide. This provides a suitable environment for examination. A few screen imagines are shown in Figures 1 through 4.

Summary

A set of questions suitable for computer-aided examination are presented. These questions are also analysed to show their appropriateness for computer implementation. The human factor consideration and the probability for anti-plagiarism are also discussed.

These questions have been implemented on a graphics workstation. Such a system has demonstrated many advantages. It is a step toward a full automated system for computer-aided education. We are looking at other types of questions to be incorporated in our existing system. Some intelligent tools will considered for future extension in this area.

References

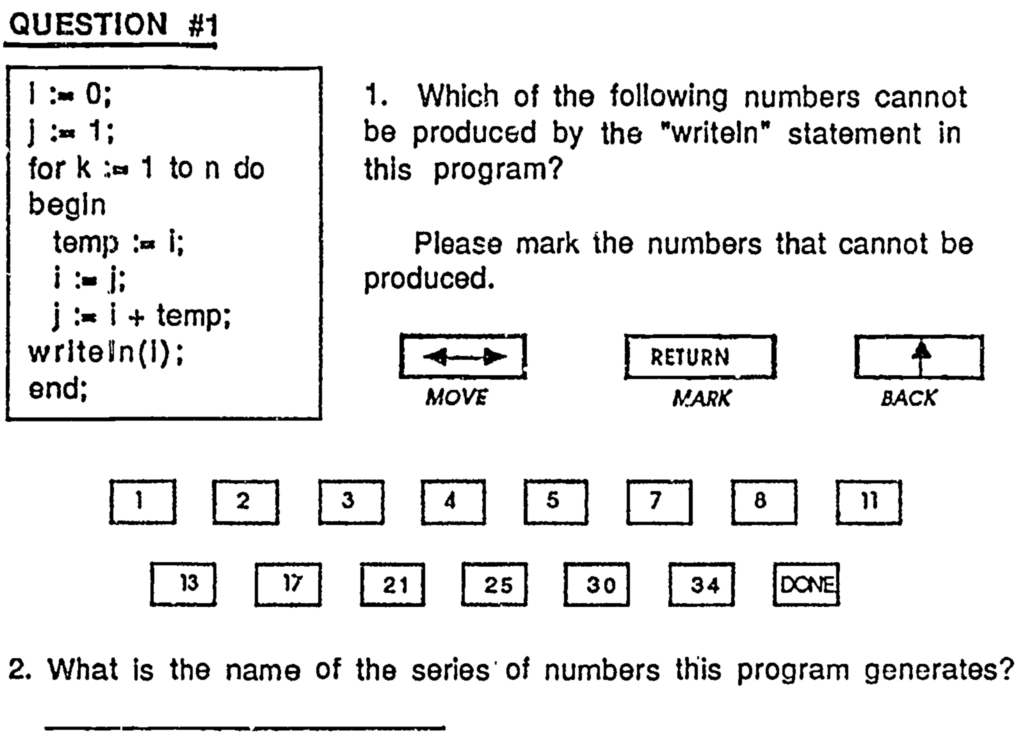
Liu, C.L. (1968), *Introduction to Combinatorial Mathematics*, New York: McGraw-Hill, Chapter 1.

Pfeiffer, P.E. and Schum, D.A. (1973), *Introduction to Applied Probability*, New York: Academic Press, Chapter 2.

Dr. Tao Li is a lecturer of Department of Computer Science, University of Adelaide, where his current research includes Artificial Intelligence, Expert Systems as well Parallel Processing.

Mrs. Luyuan Fang is now at Department of Computer Science, Flinders University of South Australia

Mr. Chongsan Yu is a visiting associate professor of Department of Electrical and Computer Engineering, University of Wollongong. His research interests include Expert systems on process control, Computer Networks and Data base management systems.



QUESTION #1

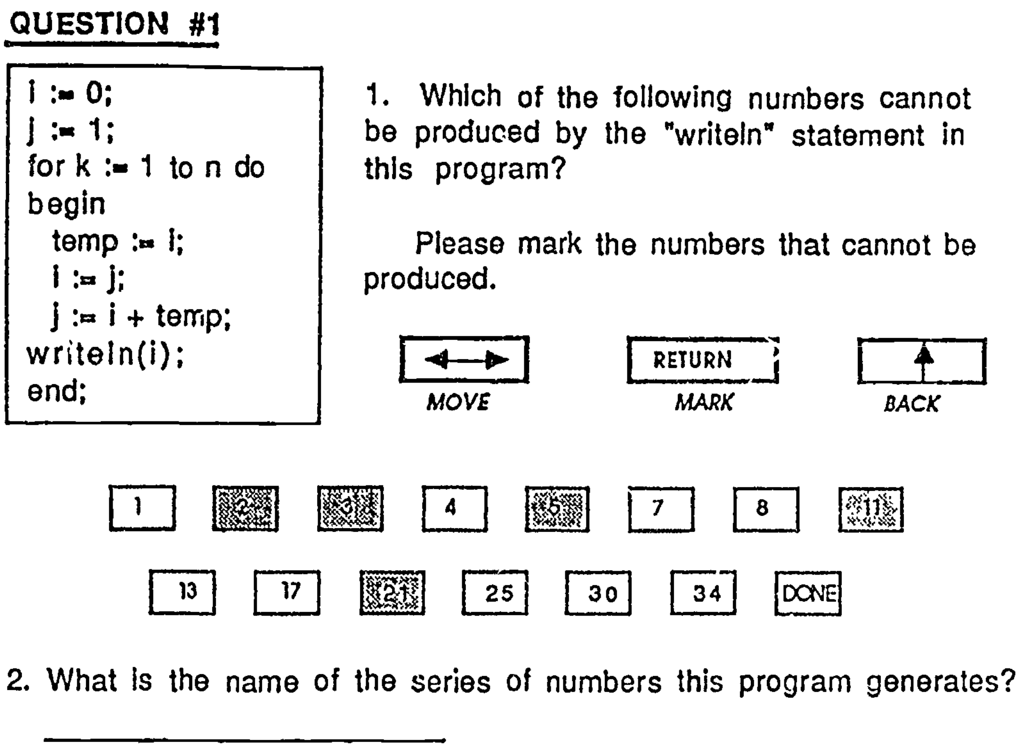
```
i := 0;
j := 1;
for k := 1 to n do
begin
  temp := i;
  i := j;
  j := i + temp;
writeln(i);
end;
```

1. Which of the following numbers cannot be produced by the "writeln" statement in this program?

Please mark the numbers that cannot be produced.

2. What is the name of the series of numbers this program generates?

Figure 1: Screen image of a multiple selection question



QUESTION #1

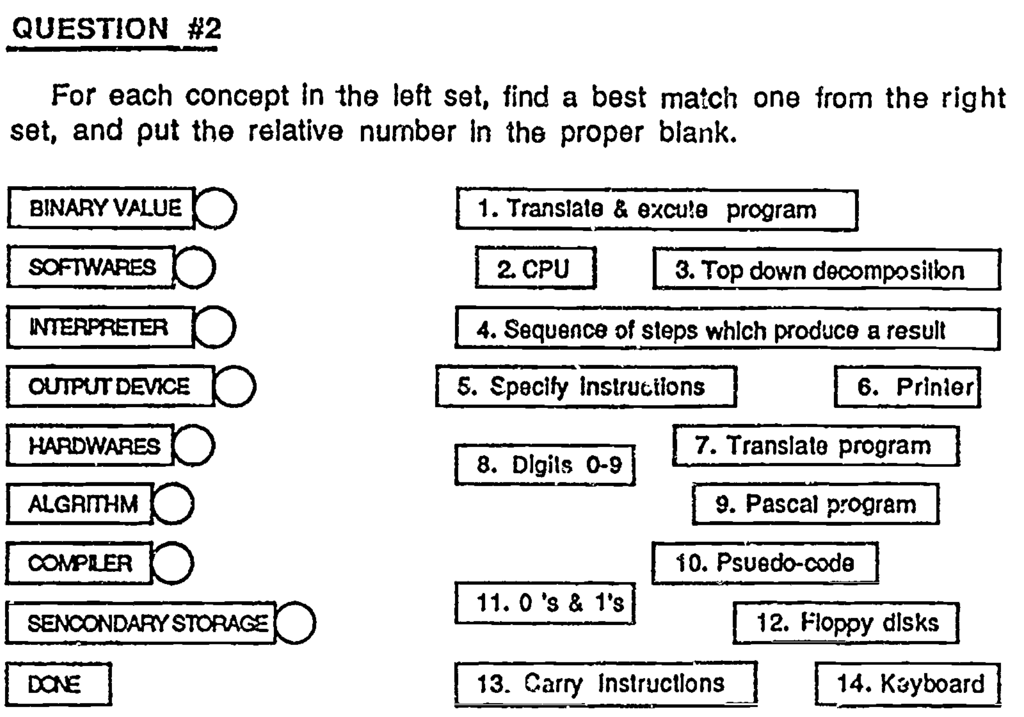
```
i := 0;
j := 1;
for k := 1 to n do
begin
  temp := i;
  i := j;
  j := i + temp;
writeln(i);
end;
```

1. Which of the following numbers cannot be produced by the "writeln" statement in this program?

Please mark the numbers that cannot be produced.

2. What is the name of the series of numbers this program generates?

Figure 2: After some choices are selected.



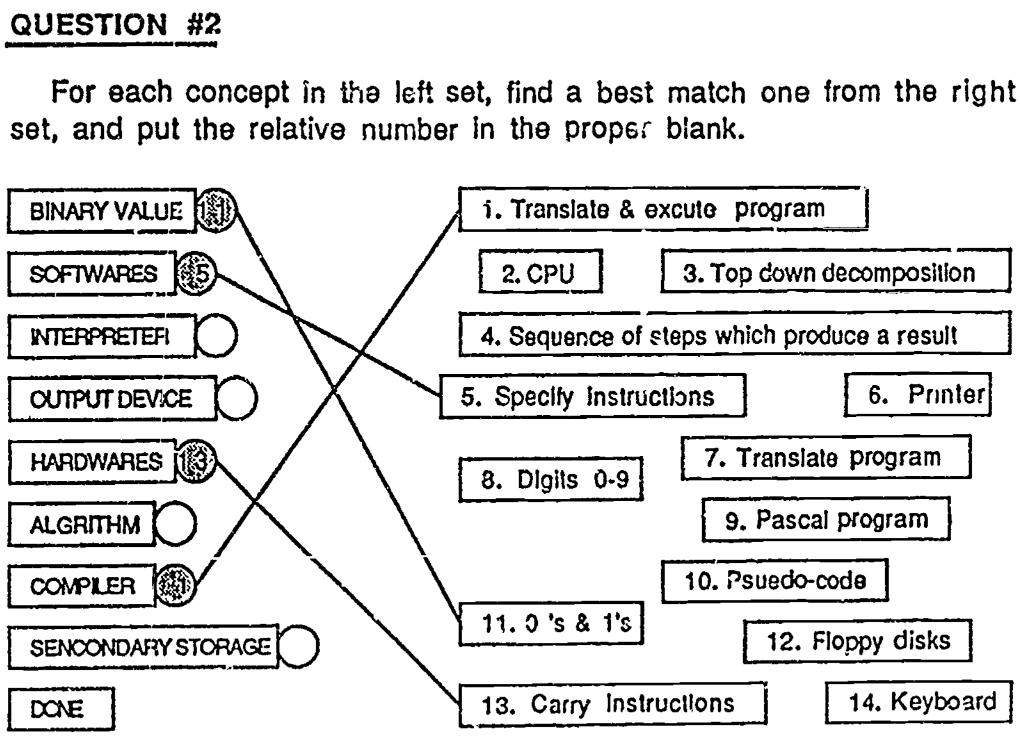
QUESTION #2

For each concept in the left set, find a best match one from the right set, and put the relative number in the proper blank.

BINARY VALUE
SOFTWARES
INTERPRETER
OUTPUT DEVICE
HARDWARES
ALGRITHM
COMPILER
SENCONDARY STORAGE
DONE

1. Translate & excute program
2. CPU
3. Top down decomposition
4. Sequence of steps which produce a result
5. Specify instructions
6. Printer
7. Translate program
8. Digits 0-9
9. Pascal program
10. Psuedo-code
11. 0 's & 1's
12. Floppy disks
13. Carry instructions
14. Keyboard

Figure 3: Screen image of a multile match question



QUESTION #2

For each concept in the left set, find a best match one from the right set, and put the relative number in the proper blank.

BINARY VALUE 11
SOFTWARES 5
INTERPRETER
OUTPUT DEVICE
HARDWARES 13
ALGRITHM
COMPILER 1
SENCONDARY STORAGE
DONE

1. Translate & excute program
2. CPU
3. Top down decomposition
4. Sequence of steps which produce a result
5. Specify instructions
6. Printer
7. Translate program
8. Digits 0-9
9. Pascal program
10. Psuedo-code
11. 0 's & 1's
12. Floppy disks
13. Carry instructions
14. Keyboard

Fgiure 4: After several matches were made

Computer Assisted Learning at the University of Otago

Marjan Vlugter, CAL Consultant / Programmer
Computing Services Centre, University of Otago

The University of Otago has made a commitment to the development and use of CAL techniques in the University. It is the first university in New Zealand where positions have been created with specific responsibilities for Computer Aided Learning: one in the educational area attached to the Higher Education Development Centre and one in the technical area attached to the Computing Services Centre. This paper will give a historical overview of CAL at Otago, describe the current situation and outline the changes that are taking place.

Over the last two years CAL at Otago University has changed. In fact, some great strides have been made and CAL at our university could be said to be growing up fast. This may be attributed to three factors:

- financial support
- establishment of advisory services
- growing popularity of the Macintosh™ computer

We will first look briefly at the growth of interest in CAL over the last two or three years. We will then look at the factors that influenced this and at the situation at the present time. Finally, we will look at some of the directions and plans for the future.

CAL then and now

In 1985 six departments had made a firm commitment to using CAL with classes (see Table 1). The Commerce Department used a computer laboratory to teach their students about the use of programs in the business world. Teaching laboratories were being set up for use in Psychology and Computer Science. Several departments were either evaluating the possibilities of CAL or ready for development work. Six departments received a microcomputer for teaching development purposes. Four of these were not yet involved in using CAL. The Anatomy Department was carrying on an experiment (started by the Orthopaedics Department) in the evaluation of interactive videotape.

Table 1. 1985

Department	*Computer*	*Type of use*
Anaesthetics Clinical	Apple II	simulations
Biochemistry	VAX	problem solving exercises
Economics	Apple II	simulations
Education	Apple II	teaching about CAL
Mathematics	Apple II	demonstrations and simulations
Microbiology	Apple II	laboratory experiments
Pharmacology	Apple II	laboratory experiments and demonstrations

By 1987 sixteen departments are using some form of CAL in classes (see Table 2). It is interesting to note that of the 9 new users, 6 use the Macintosh and that 2 of the older established users have switched to or have added the Macintosh. The Physiology Department has replaced their oscilloscopes with a laboratory full of Macintoshes, using locally developed, very sophisticated, software. There are plans to extend the use of the Macintoshes from laboratory tools to incorporate CAL exercises as well. There are similar plans in the Physics Department. It is expected that by this time next year another eight or ten departments will be using CAL applications and that those who started off carefully and conventionally will have ventured out in more diversified use of CAL. The reason for this growth should be mainly attributed to the three factors mentioned above and we shall now look at each of these factors individually.

Financial support

Because it was noticed that there was an increase in applications which expressed an interest in developing Computer Assisted Learning units and which requested computer equipment for use in teaching, a working party was formed in 1984 in order to obtain an overview and evaluation of current and possible future activities in CAL and to advise on the implications for funding policy.

The working party visited 20 departments and groups within the University of Otago and reported in June 1985 that there were sufficient interesting developments of a practical sort to warrant a modest increase in support for CAL in the University. Three levels of support were suggested:

- expert advice and information in the early stages

Table 2. 1987

Department	Computer	Type of use
Anaesthetics	Apple II	simulations
Anatomy	Macintosh	drill & simulation
Clinical Biochemistry	VAX	problem solving exercises
Dentistry	Macintosh	drill
Economics	Macintosh	simulations
Education	Apple II	teaching about CAL
French	VAX	drill
Geology	Macintosh	numerical analysis simulations (spreadsheet)
Mathematics	Apple II &	demonstrations and Macintosh simulations
Medicine (Wellington)	VAX	drill
Microbiology	Apple II (more units)	laboratory experiments
Pharmacology	Apple II	laboratory experiments and demonstrations
Physics	Macintosh	simulations & concept learning
Russian	Macintosh	drill
Textiles	IBM	drill and tutorial
Zoology	Macintosh	drill and simulations

- practical support (hardware, programming assistance) when ready to proceed
- establishment of CAL units (consisting of 5 - 10 machines) when ready for class use

The working party also made the following recommendations:

- the appointment of an educational adviser and a computer programmer to give advice and assistance in CAL
- provision of some out-of-hours CAL computing facilities for students
- a one day exposition of CAL to foster communication among people interested in CAL

It is remarkable that each one of these suggestions and recommendations was accepted. An exposition was organised immediately in the winter of 1985. For many people this was the first occasion to become aware of CAL activities at the university and many of those working with CAL became aware of the others for the first time. Out-of-hours computing facilities for CAL were offered by adding specific machines to the microcomputer laboratory in the Computing Services Building. The appointments of an educational adviser and a programmer were approved. The working party became a permanent sub-committee which now makes recommendations on all applications relating to computer support for CAL. Because it was considered impossible for the sub-committee to address all educational uses of the computer, it decided at an early stage to concentrate on one specific application: Computer Assisted Learning.

Establishment of Advisory Services

In the winter of 1986 the two CAL advisers commenced work, one in the educational area attached to the Higher Education Development Centre and one in the technical area attached to the Computing Services Centre. The major objectives, identified by the CAL sub-committee, were:

- fostering of communication between departments with an interest in CAL
- ensuring the educational soundness of the design of CAL applications
- encouraging the efficiency of software design

Fostering of communication

In the early stages many departments or solitary persons in a department had worked in isolation without being aware of what others were doing. The first CAL exposition in 1985 soon made it clear that in some cases efforts had been duplicated, been needlessly laborious or that some people had been 'reinventing the wheel'.

To overcome this problem it was decided to form a "Software Society". This group meets several times a year to discuss areas of CAL use, design and development. Meetings may consist of demonstrations, presentations, discussion or the exchange of technical (programming) information. Seminars and lectures publicised among the wider university community, aimed both at regular users of CAL and at those people who are still in the information gathering stage, are also given several times a year.

The consultant/programmer attached to the Computing Services Centre acts as a central advisory service and clearing house of CAL related activities and information. Anyone seeking information or advice regarding CAL will in the first instance contact one of the advisers. Depending on the stage of development, it will be the task of either the educational adviser or of the programmer (and sometimes of both) to keep in touch with and offer assistance to the department in question.

Educational soundness of CAL applications

It is primarily the task of the educationalist to advise people on the educational implications of CAL, on the effects it may have on the curriculum, on the various ways CAL could be approached educationally, on the educational aspects of software design and on the evaluation of CAL use. Where a department is contemplating or planning the introduction of CAL for the first time, the educationalist and the programming consultant will work together with representatives of the department, locate and evaluate or design and program the software to be used, offer advice on the financial implications, where feasible set up a pilot study and evaluate the project.

More efficient software design

The earliest programs used in CAL were often programmed locally by the teaching staff, generally written in BASIC, usually run on the Apple II computer. In most cases, those involved with the programming felt that the task was too time consuming and too distracting from teaching and research. It was hoped that the appointment of a professional programmer would make it possible to develop more software more efficiently.

The results, a year later, may in some ways be considered extremely encouraging, in other ways disappointing. In 1987 there are nine more departments using CAL exercises. However, seven of these almost exclusively use simple drill and practice exercises. There are some very specific reasons for this:

- software design: a time consuming task. The most vital part of the development of a CAL application is the design of the software and the integration of the use of it in the curriculum. This is a task that has to be performed by the teaching staff, even though the programming itself does not have to be. As a result less software has been developed than had been hoped, because the time and effort required from the staff was not available or was not seen to be sufficiently worthwhile.
- one-off applications: are they cost effective? People are becoming more aware of the costs involved in CAL, both in hardware and in person time. Since the benefits are only rarely or sporadically evaluated, it is hard to justify the investment. On a personal level, teaching staff can always be assured of the results of time invested in research, not of time invested in unevaluated attempts to improve teaching.
- The Macintosh: a friendly machine, but difficult to program. Since its appearance, the Macintosh has been seen as an ideal computer for CAL and at Otago the use of this computer in the CAL area is overtaking the use of others. In the next section we will look at the reasons for this and the consequences for the CAL programmer.

Growing popularity of the Macintosh computer

The main reason for the great popularity of the Macintosh lies in the very carefully designed user interface: "The Macintosh is designed to appear to an audience of nonprogrammers, including people who have previously feared and distrusted computers." (from The Macintosh User Interface Guidelines, Inside Macintosh, page I-27). In general, developers of Macintosh programs follow the guidelines given by Apple and the result is consistency, so that users do not have to learn a completely new interface for each application. The learning time for using the Macintosh is very brief, some say non-existent, and this is particularly important in CAL were large numbers of mostly naive computer users have to become familiar with a program they may use only two or three times.

However, the Macintosh is considered one of the hardest machines to program and even for an experienced person, learning how to program 'the Macintosh way' may take three months or more, assuming one is already fully conversant with programming itself. This fact does not add to the cost effectiveness or ease of development of CAL programs! At present four departments are developing applications for the Macintosh and Macintosh expertise is gradually building up.

On the plus side is the fact that in the United States the Macintosh has been adopted by many universities as the ideal machine for use with students and a steady trickle of software suitable for CAL purposes has resulted from it. At the same time an increasing number of very imaginative programs is becoming available, which are suitable for use as authoring or software tools for CAL units.

Another approach, and one followed consistently by the Mathematics Department, is to use standard modules and to develop ;eful modules and development tools firs, as building blocks for later specific developments. A modular programming language such as Pascal is most suitable to such an approach and since Pascal is the 'native language' of the Macintosh, modules are developed in that language. A large range of modules, some most specifically useful for mathematical manipulation, has now been developed.

The present situation

As a result of the factors outlined above, the CAL programmer/consultant has spent less time actually writing application programs and more time trying to locate and encourage the use of existing CAL programs, software development tools and more efficient design methods.

One of the problems with existing, ready-made programs is that they are seldom just right. The most often heard comment is: "Yes, it is a really nice program, but.... and if only...". For this reason the most effective programs are authoring languages or authoring tools. Since authoring languages are the easiest to use and require the least input from the teaching staff (sometimes nothing more is done than the typing in of old exam questions by a typist!) they tend to be a first choice.

However, more and more suitable software tools are coming on the market and potential CAL users are becoming more aware of the possible uses of tools such as spreadsheets, data bases, modellers. Comments have been heard from one or two departments, which at present mainly use the computer for drill exercises, that the limitations of drill exercises are becoming irksome. Now being totally familiar with the computer and some of its potential, there is a strong tendency to branch out. Three or four departments are using spreadsheets on the Macintosh (Excell, Multiplan, Jazz) for CAL exercises and more departments are at present experimenting. The program STELLA, a modelling program for the Macintosh, has been slow in taking off, but is generating a growing interest as a simulation and modelling tool. Animation and pictorial databases are other directions currently under investigation.

There is a policy to buy software suitable for CAL centrally, so that it can be evaluated by the CAL advisers. Generally, new programs are announced in the Computing Services Centre Newsletter, so that departments may request demonstrations or a loan of the program for evaluation before deciding to buy it. Those known to be interested in a certain program are personally contacted. In general there is an atmosphere of cooperation and information sharing and the result is that efforts are less likely to be duplicated.

The educational adviser has lectured and given workshops on the educational implications of CAL. He has been encouraging the careful planning of projects, the use of pilot studies and evaluation of the exercise, using recognised research methods, thus making it possible to publish the results and to consider this work part of ongoing research. The underlying objective of course is to evaluate and possibly improve the effectiveness of CAL and its overall place in the curriculum.

It is becoming harder to define CAL, as people become more familiar with the use of microcomputers as tools in everyday activities. When is the user learning and when is the user merely addressing the computer as a tool? The CAL sub-committee may in future be less able to pin down CAL and be compelled to broaden its criteria. It has been suggested to refrain from using the abbreviation CAL, and rather refer to "the use of computers in teaching and learning": UCTL. Quite a mouthful and it has not as yet replaced the easier term CAL.

Some future directions

CAL has sometimes during its history been seen as a time saver, a personnel saver, a panacea, or simply the bandwagon to get on as fast as possible. Although occasionally such thoughts do seem to be entertained by some of those showing an interest in CAL at Otago, the centralized and structured approach to CAL usually manages to identify and to dispel them. The growing interest in CAL is therefore in general accompanied by a careful consideration of all the factors involved, educational, technical and financial.

The last factor especially will be the deciding one in whether or not the current trend of a steady growth in the number of departments using CAL will continue. There is a tendency to develop and evaluate CAL with small groups at first, often second or third year students, and then to gradually extend the use of CAL, as it is found effective, to the larger classes. The logistics of this will have to be carefully thought out: whether it is feasible to accommodate large enough computer laboratories in each department using CAL or whether large public access laboratories, possibly networked and requiring management, might be needed. This question of access to hardware effectively forms the ceiling of future growth in CAL.

Some developments that are expected in the near future and some of the investigations currently going on:

- It is expected that more departments will turn away from authoring languages and drill exercises and that the use of 'application building programs' (such as spreadsheets, modelling programs, animation, hypertext, data bases) will increase.
- The University will be investigating the use of Interactive Videodisc in CAL. Funds have been set aside for this purpose.
- An area of research is the use of Software Engineering techniques for the design and development of software.
- Discussions are going on for the establishment of a pilot study for the use of a set of computer writing tools for an English writing skills course
- Because of the benefits seen in a central point of information it is considered important to set up a national contact group for tertiary CAL in New Zealand.

Summary

In the early days of CAL at the University of Otago departments were working in isolation, often on an ad hoc basis. Most programs were developed locally, written in BASIC for the Apple II. Because of a centralized approach, starting off with a university wide investigation and followed by both financial and advisory assistance to departments wishing to use or develop CAL, the interest in CAL has increased steadily and the approach to CAL has matured, become more structured.

The three levels of support recommended by the CAL sub-committee have been operating successfully: initial advise and analysis of the needs followed by allocation of a development machine together with programming and educational advice and assistance and finally making available a set of microcomputers, allowing the introduction of class use.

The writing of stand-alone applications has decreased. The use of ready made programs or programming tools has increased. The approach to programming has become a modular one and Pascal is the main language used. The use of Software Engineering methods in program development is an area of research. Gradually the Macintosh ™ is becoming the preferred CAL machine, because of its 'user friendliness'.

A department contemplating the use of CAL is carefully accompanied in its investigations with advice. Where possible a pilot study is set up and increasingly the introduction of CAL will be evaluated and the results used to improve the effectiveness and the appropriateness of it in the overall curriculum structure.

A Computer Based Testing and Evaluation System

Bill Chia: Computer Centre
and Mon Khamis: School of Education
Nepean College of Advanced Education

The Design and delivery of many Tertiary Education courses are based predominantly on traditional presentation methods using lectures, seminars tutorials and reading materials. However, there are other techniques available to support these courses.

A computer based testing and evaluation system has been developed on models which can incorporate techniques such as diagnostic feedback, decentralised learning, self testing, monitoring or progress and mastery learning. The system can provide reports to support summative evaluation, test analysis and item analysis. Furthermore, the data can be presented in a form that supports ongoing formative evaluation of the course on which the test is a part.

The approach adopted in developing this computer bases system has been to make it straightforward and powerful enough to be used by teaching staff who may have limited experience with computers, yet sufficiently general and flexible to meet the needs of different subject areas or different teacher's goals. Within the confines of the system, the individual student can be given some degree of control. The system can be adjusted to give guidance to the student on the basis of performance. It may readily be expanded to allow other facilities to be incorporated.

The system was developed with the aid of the PLANIT authoring language and has been applied to subjects in the Schools of Business, Nursing and Health Studies, and Education. The paper describes the system and its application to a course in Research Methods in the School of Education at Nepean C. A. E. Implications for the design of courses at the Tertiary level are discussed.

Many Tertiary Education courses are based predominantly on traditional presentation methods using lectures, seminars tutorials and reading materials despite the fact that there are other techniques available which could support and enhance the teaching of these courses. Computer based systems are one of the possible range of techniques.

Over the last few years, there has been considerable discussion and consideration given to methods of development of computer based teaching and learning materials. Some have argued for retaining the flexibility of General Purpose computer languages such as Pascal, while others have pointed to advantages of specialised authoring languages. Yet it has also been pointed out that even the authoring languages provide too little in the way of higher level constructs (Merrill 1985). Authoring languages may be valuable to authors if they lack programming skills and have too little time to gain them. Systems involving higher level constructs would be helpful to them and to any who have limited experience in instructional design.

Authoring systems, as distinct from authoring languages, may provide some suitable models, though they are often only available on specialised hardware. The majority of academics in tertiary education are likely to find that their needs aren't met very well by the materials they can readily obtain. Commonly, the materials differ in emphasis from that desired and often require a particular brand of hardware which may not be available and cannot be justified on the basis of needs of one part of one course. Most would be daunted by the prospect of doing their own programming and would not have the time anyway.

At Nepean C.A.E. a system has been developed around an existing Authoring Package (PLANIT) to provide simple-to-use computer based assessment and evaluation. The resultant system has a number of the features that a range of academic staff would find useful. With these features they can get the benefits of several types of computer based testing without the need to learn a programming language or an authoring language.

The system makes use of options which can be selected by the Academic user and applied to their own course materials. The individual ideas in the system are relatively simple, but they help to bridge the gap between what a lecturer may wish to achieve with the aid of computer, and the level of practical programming knowledge or skills which he or she may currently possess. This system allows easy development of some computer based assessment and evaluation procedures for enhancing learning. It can be used as part of ongoing assessment or for final assessment.

Lack of programming skills may be one reason that academics might avoid computer based support for their students. Limited knowledge of instructional strategies and instructional design may be another. The system we have developed allows the combination of various optional features to produce a number of coherent testing models. The options may be set specifically or patterns of options may be accepted by default. The user interface is a set of menus which feed the information through a 'preprocessor' to generate PLANIT Language 'lesson code'. This lesson code is quite quite easy to understand and can, if desired, be modified through a standard editor or through the PLANIT System itself. The lessons, however, usually run satisfactorily without modification. In this way, the power of 'high level decisions' is readily combined with the flexibility of 'low level' detailed adjustments.

The system has been applied to a range of subject areas. It has been used for Behavioural Science, Management Studies and Auditing in the School of Business; Measurement and Evaluation, Research Methods, Educational Psychology in the School of Education; Human Growth & Development and Behavioural & Social Science in the School of Nursing and Health Studies. A set of tests in biological science are currently being developed.

Description of the System.

Test items, in the form of multiple choice questions, are gathered and arranged into groups on the basis of subject matter, difficulty level or some other criteria. These questions become the basis of a kind of "on-line" test bank. From this bank of items, questions are selected according to a pattern as decided upon by the academic user. The items are made available to the student via a centralised computer. The student can take the test at his or her own time (within a predetermined period, at selected times as decided by the academic). The academic can retain much of the control, or make it available to the student as he or she thinks fit. Options such as "immediate feedback", "progress reports", " printed report or certificate" can be decided upon by the academic. A most important aspect associated with assessment and reporting relates to its use for "measurement of progress" and "mastery of content". The advantages which are available to foster and enhance learning can often be lost or reduced because of the difficulties of providing the appropriate feedback and guidance to the student. By means of the system reported here, tracking of student responses and performance data within the various subject matter areas or various difficulty levels can be made available on request. Results for an overall test or group of tests is gathered automatically and can be retained or discarded, as the academic may require.

How to use the system

The academic users carry out the following steps.

1. Provide the questions, together with an indicator for the accepted answer or answers.

2. Group the questions by subject matter areas, difficulty levels, educational objectives, or other criteria.

3. Decide what testing model and options are to be utilised, and what access is to be made available to the students.

4. Decide what reporting functions are to be carried out when and if required.

5. Trial the resultant test

6. Inform the students of the test's availability, any special conditions, and how to access it.

7. Obtain reports and use accordingly.

Services Provided

The services provided for the academic are:

- Data entry of the 'raw' question items.
- Conversion to PLANIT Language formats
- Delivery of the test to the student.
- Gathering of report data
- Reports to the student: this includes progress reports, end-of-test reports and summarised results.
- Reports to the lecturer: This includes detailed reports (if required) and reports summarised by class, by topic or by course.

Of the above, the data entry is provided by the Computer Centre staff; the rest is provided by the system.

Application of the system to an Educational Research Methods Course

Formative and summative methods have been used to assess teacher education students in an educational research methods course. Traditionally, the formative exercises comprised of workshop reports and a major assignment. The summative procedure was an end of semester exam. These evaluation procedures did not provide students with appropriate motivation, and the necessary feedback on performance to plan ongoing study strategies to master the course content. The students have viewed these procedures as major obstacles to be overcome at certain times in the semester and did not provide a positive structure for the average students to assess their level of performance and plan study activities to compensate for deficient areas of knowledge.

Certain features of the computer based evaluation system were used to develop a formative assessment exercise to overcome some of the criticisms of the traditional class tests and to provide students with an opportunity to obtain feedback in a non threatening environment. This feedback allowed them to identify areas of course content they were familiar with and other areas needing further study,

Structure of the class test

The educational research methods test contained a bank of 50 multiple choice items with four possible options for each item. The items were grouped into five areas. The students completed 25 questions randomly chosen from the 50 item bank as follows:

1) nature of educational research
   8 questions - 4 chosen
2) hypothes testing
   10 questions - 5 chosen
3) data collecting procedures
   14 questions - 8 chosen
4) scaling procedures
   7 questions - 3 chosen
5) interpreting statistics
   11 questions - 5 chosen

The correct response for each question was incorporated into the system in order to provide the student with feedback on test performance. The students obtained an analysis containing the total number of questions attempted, the number of correct, and incorrect answers and their respective percentages.

The students attempted each question chosen once. They were not permitted to choose another option if their first choice was incorrect. However, this would be a useful option within the test structure. Students could have a second attempt, and if this was incorrect then the correct option would be given.

The time element was not considered a major factor in the development of this class test. Students had two minutes to respond to each of the 25 items chosen. However, the time element can be useful learning stimulant, and can be adjusted according to the number of attempts made by each student or students can be given an opportunity to adjust the response time to suit themselves.

The item bank of 50 questions was very limited. This could be substantially increased in order to test content areas more exhaustively, particularly if students attempt the test several times. An available feature which was not incorporated into the test is the analysis of scores for each section of the test and the provision of feedback on performance in each area. The student can then be directed to read certain chapters of the text relating to the items the student got wrong.

The level of mastery was not incorporated into the test. A useful feature would be to set a mastery level of say 75%. If the student did not achieve this then he would be directed to learn certain areas of content before retaking the test.

The diagnostic procedures are useful to students and course presenters. If certain items were causing difficulty to students then the concepts associated with these items need further reinforcement and clarification. Lecturers can ascertain certain areas of knowledge which cause students more difficulty and plan the course to cope with this.

The non threatening environment can be a major motivating factor in student learning. If students are given the opportunity to undertake the test a number of times until they have achieved a certain level of performance then they are more likely to continue to learn because no judgment is made on their performance by the lecturer or their peers. A further modification of the mastery process may include a barrier for each section of the test. The students must achieve a fixed mastery level in each section of the test prior to any new items being presented.

The options can be included in the system so that if certain questions containing basic concepts were correct then extension questions could be presented. Unless this occurred then no new questions would be presented and the student would be instructed to study sections of the text prior to retaking the test.

These options allow the student to practice exam strategy and identify strengths and weakness. The student would develop an overall strategy to sit for the final exam.

The course could be modified to reduce the impact and the threat of summative evaluation and change the orientation to informative evaluation. The options available in Plan should be used to allow the student three attempts within a given period of time. The results of the three attempts could be aggregated and converted into a final score for the summative evaluation. The benefits which could accrue from this procedure relate to the conversion of short term memory to long term memory. Students are more likely to recall information if they have an opportunity to learn it for mastery and understanding rather than recall. This could be achieved by modifying the test structure to change the correct option for each item. This may prevent the student recording the question and its correct response, and force the student to develop an understanding of the concepts associated with each question instead of recalling factual responses.

Appendix: Models and Options

There are a number of models which have established their usefulness. These form some of the components available for use within a course.

- MASTERY LEARNING
- FORMATIVE AND SUMMATIVE ASSESSMENT
- DIAGNOSTIC FEEDBACK
- STUDENT CONTROL
- SELF TESTING
- MONITORING AND ADVISING
- REPORTING ON THE BASIS OF CONTENT, DIFFICULTY, INDIVIDUAL, CLASS and COURSE.

The facilities which have been made available may be thought of as TOOLS for constructing models. These are available in the form of OPTIONS which allow specific setting of various alternatives to suit the teacher or learner's particular goals. The options may be used for the assessment system or for test-cum-tutorials which provide some extended support material. Some of these OPTIONS are available interactively from a menu, others are provided by making a few adjustments to the lesson file. Options currently available or proposed, are set out below.

OPTION: Feedback or none
DESCRIPTION: Right-Wrong feedback,
with or without Answer Correction,
with or without Answer EXPLANATION for some or all questions,
with or without Specific Feedback on particular answer chosen,
with or without Branching (increases complexity).

OPTION: Immediate Retry
DESCRIPTION: variations are:
Not allowed, permitted, or forced (1, 2, 3 or more times)
POSSIBLE USES: Tests, Self Tests, Mastery

OPTION: Time Limit (on or none)
DESCRIPTION: Maximum time allowed per question. May be
set by author
set by student (if permitted by author)
set by lesson performance

OPTION: Random Order Presentation.
DESCRIPTION: The questions may be presented in random order.
They can also be randomly selected from defined subgroups of questions. The number to be chosen from each subgroup can be set by the author.
POSSIBLE USES: Generate a different test for each student.
The choice could be set by the student or by lesson performance criteria.

OPTION: Progress Function.
DESCRIPTION: Gives a summary of student's progress (No. seen, right, wrong, percentages).
During and after test, or at end only (author's choice).

POSSIBLE USES: Extend to give advice to the student. Advice may be based on sub-categories of questions such as content, specific objectives, difficulty etc.
SUITABILITY: Tutorials or self test.
EXAMPLES, RATIONALE: Guide or motivator.

OPTION: Finish at any point.
DESCRIPTION: Available at the option of the author. Can also specify, resume at the same point or resume at the start, or specified points only.
POSSIBLE USES: Allow or withold student control. Depends on material

OPTION: Retake the test
DESCRIPTION: Immediately or after a given period (unlimited or restricted number of times, or disallowed)
POSSIBLE USES: Self Test or mastery testing or assessment.
SUITABILITY.EXAMPLES, RATIONALE: Student control, or not?

OPTION: Printed Certificate or Report.
DESCRIPTION: Prints a Certificate or Report if preset score or percentage ranges are attained.
POSSIBLE USES: Extension to give subgroup scores.
SUITABILITY, EXAMPLES, RATIONALE: Mastery, self test goal setting and "reward".
Provide written advice to student.

OPTION: Branching and Return.
DESCRIPTION: Branch to sections which can be selected interactively.
POSSIBLE USES: This allows review or preview of test material. It can also be used to provide supplementary information, Hints, Data, Worked examples, context sensitive advice or help.

OPTION: Branching (No automatic return).
DESCRIPTION: This allows branching to a (sub)section or around a (sub)section. It may be placed under author control, student control or student performance control.
POSSIBLE USES: Remedial or fast track.

COMBINATION OF OPTIONS: Options may be combined to produce particular environments for learning.
e.g. GROUPING BY DIFFICULTY, PROCESS
e.g. GROUPING BY CONTENT, ... REPORTING.

References

Merrill M. D. (1985). Where is the Authoring in Authoring Systems? *Journal of Computer-Based Instruction,* Autumn, 12(4), 90–96.

Mevarech, Z.R., Time Engagement and Achievement in Computer Assisted Instruction, *Educational Technology*, July 1986, 38–40.

Bill Chia is a principal tutor (computing) at Nepean College of Advanced Education. His current position is Academic Computing Advisor and Liaison. He is a member of the organising committee for the CALITE-87 Conference.

Dr. Mon Khamis is a lecturer in the School of Education at Nepean College of Advanced Education. His current research interest is in the area of "Competencies for the successful classroom teacher and the successful school Principal".

Profile of a successful CAI facility — Identifying useful technology

Graham R Parslow, Biochemistry Department
T Robert Haynes, Advisory Centre for University Education
University of Adelaide

Success in CAI (Computer Assisted Instruction) depends on the use of appropriate technology, the selection of suitable material for presentation and management which allows adequate student access to courseware. The Biochemistry Department at Adelaide University has been using CAI for a decade with gratifying results while spanning 3 generations of personal computers. The present Biochemistry Department facility of ten Olivetti M24 personal computers is able to serve the course revision requirements of over 300 students who use the facility voluntarily. The principal teaching tool is the "Q" Instruction Package serviced by a data base of some 9000 questions (the Medical Biochemistry Question Bank). At the pioneering forefront, optimistic predictions can be made for videodisc applications and the authors will discuss their involvement in the production and application of the first Australian University videodisc.

The reason that CAI deserves serious attention is that it is a medium which reaches a majority of students in a manner that is different from other teaching contacts and technologies. Saving teaching money or liberating staff time are not valid arguments in support of CAI. As with other teaching approaches, the essential ingredient for success is a belief that CAI has a valuable contribution to make. Education specialists look for absolute evidence of pedagogic value, but the case is very much that you will get out of CAI what you believe you will.

In the oral presentation we will illustrate our views by reference to the hardware and software we have been associated with. We will speculate on the applications for new generation mass storage devices, particularly in regard to video images.

What do we learn from technology extinction?

Technology which is fated to survive is responsive to individual interaction and powerful enough to perform truly useful tasks.

Novelty is an undoubted component of short-term success. Sputnik aroused amazing interest, although it was almost functionless. It is difficult in the early days of a technology to decide whether state of the art technology is mature or simply a portent of truly practical machines to come. The authors are of the view that contemporary personal computers and video disc technology will continue to contribute to useful teaching.

In the early 70s colleagues at Melbourne University Biochemistry Department documented the successes of their on-line facility for undergraduates to use a teleprinter terminal to analyses enzyme kinetics. In the Adelaide Biochemistry Department Sinclair ZX-80 in the late 70s fascinated students who saw it translate genetic code sequences into protein structures, all in 4K of RAM. The idea of using this equipment today is laughable, but it does illustrate the motivational value of using what is perceived as state of the art hardware. Novelty in these cases obscured long-term utility.

Those of us who rode in on the pioneering wave of personal computers in the late 70s and early 80s thought that Apple IIs, Commodore PETs and BBC microcomputers would perform each and every educational task capable of imagination. It has not proved to be so, the 8 bit computers of the early eighties have been relegated to the status of transitional technology. The pioneer personal computers set new standards for user friendliness however, and this has been their legacy to present generation computers.

The teaching suite of Dr.Robert Beynon (Liverpool Biochemistry) has built an enviable international reputation for CAI using Commodore PETs and Apple IIs. This work has been negated completely by changing to a network of contemporary MS-DOS computers - an entirely new teaching software base is being created. This story mirrors the demise of Exidy Sorcerer computers in the Biochemistry Department at Adelaide University.

Which aspects of teaching technology can survive the hype of immature expectation?

The large scale funding of unsuitable technology has done long term damage to general funding prospects for teaching. The flirtation with educational television is all but dead. Television monitors droop dustilly towards disinterested audiences in lecture theatres and teaching laboratories around the world (many of them still Black and White). Mass oriented educational television lacks the ability to cater for the pace at which an audience is responding to the message and has suffered accordingly. An apparent success in the use of television is the real-time link between London Medical Schools for teaching in which multiple screens and camera options are used in a "tele-conferencing arrangement". This perceived need for some aspect of one-to oneness is epitomized by disastrous attempts to use shared listening stations of six head-phones linked to a single cassette player in the Biochemistry Department at Adelaide. This head-phone share arrangement was a solution to the then high cost of the technology, and encouraged by the enticing glossy from the manufacturer showing happy students in an atmosphere of mutual learning ecstasy.

The tape-slide teaching facility in the Adelaide Biochemistry Department is a survivor of technological approaches to education. It still fulfils a valuable role in non-lecture teaching because tapes, with accompanying written handouts, can impart a higher density of new information than computer screens. The cassette players used for teaching no longer represent novelty in technology, but an everyday machine.

The elements of cassette tape which commend it include

- one standard - a cassette tape can be shared internationally
- it is personal - cheap enough for 1 tape player per person.
- it is unintimidating in its controls.

Computers have not reached this degree of establishment and are therefore duly regarded with scepticism for educational applications. The wait and see approach however is self defeating - the time to enter the learning curve for computer applications is now when computer technology is reasonably affordable and adequately powerful.

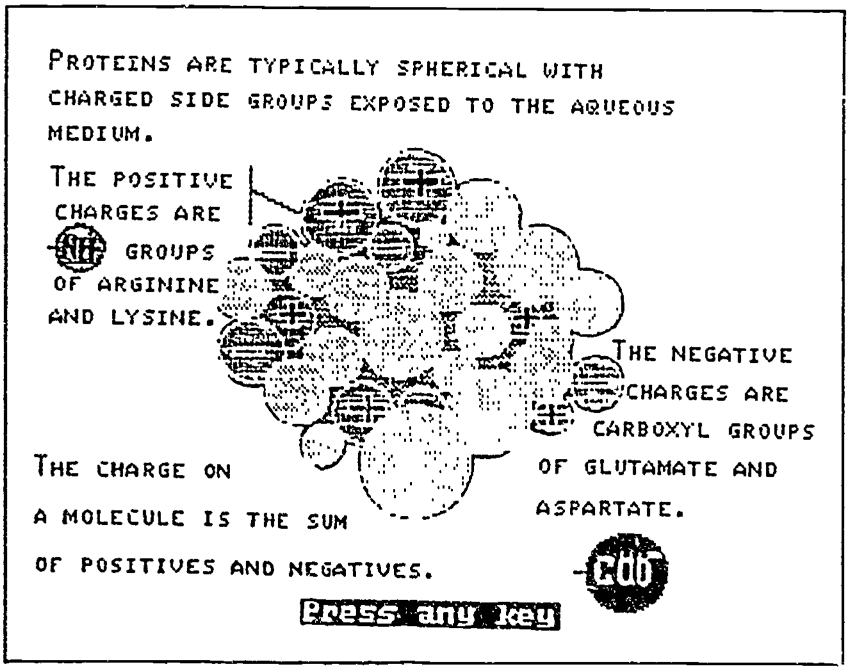
How powerful do educational microcomputers need to be?

Computer power is not a real issue. The essential question is "will the computer allow me to extend my teaching at a level which is satisfactory for my courses." Technical capabilities of the machinery are increasing logarithmically, but our educational requirements are not. We now have affordable machines that have adequate speed, memory and physical compactness to be useful.

Contemporary powerful computers (main frames) are not an answer to any real educational problems in our experience because they are made by computer-in people for other computer-in people. Powerful computers are usually expensive and begrudgingly available from their managers. For students big computers mean rigid scheduling and limited time of access. An intimidating Big-Brother log-on procedure then exasperating delays. If a student spends ten minutes pondering the truth of a statement, the expectation is none-theless that the computer will respond instantly when a key is pressed. For teachers there is the need to either master difficult authoring software or master the technical resource people who have.

Not powerful enough is disastrous too.

The UK has retarded its development of CAI by so whole-heartedly adopting the BBC micro. Nigel Gardner of the Computers in Teaching Initiative (CTI) in the UK has succinctly argued this case and it is a compelling conclusion. The grounds for this are technical and human. Technically the BBC computer is, in its most common configuration, incapable of more than simplistic modelling and shallow gimicry. The commitment of funds has been so extensive that the coffers are bare for re-equipping with better technology. In Australia at least, funds have not been dissipated by any initiative of the Federal Government to introduce computers to education.



PROTEINS ARE TYPICALLY SPHERICAL WITH CHARGED SIDE GROUPS EXPOSED TO THE AQUEOUS MEDIUM.

THE POSITIVE CHARGES ARE NH3 GROUPS OF ARGININE AND LYSINE.

THE NEGATIVE CHARGES ARE CARBOXYL GROUPS OF GLUTAMATE AND ASPARTATE.

THE CHARGE ON A MOLECULE IS THE SUM OF POSITIVES AND NEGATIVES.

Figure 1: A screen from a "Q" session computer tutorial to teach aspects of pH. Computer tuition on this subject exploits the ability of a computer to interact with students with unfailing patience and consistency.

How many students should work simultaneously at a station?

Assuming voluntary self-paced usage it may be one, two or four. One user only per station is appropriate when the material is new or difficult, otherwise with multiple use the advantage of freedom from peer judgment is lost. Even the best of friends may not work well together at a computer if their talents are unequal. Groups of four are quite successful for medical students taking revision questions together. Medical students as a group have characteristically high social cohesiveness plus academic competitiveness and enjoy audiences to their performance (at least when they are scoring highly). This argues for keeping enough space between stations to allow people clusters to form (regrettably few places of learning are built with suitable space).

What material is best presented by computer?

The mindlessly repetitive tasks of teaching are best suited. For example a tutorial on the Biochemical importance of pH at Adelaide begins by asking:

> When pH is high, is H+ concentration
>
> 1. High
> 2. Low ?

This low-brow question is insultingly basic when asked by a human tutor (despite the widespread ignorance of the answer). This computer tutorial then progresses through some elementary mathematics of important chemical principles. When human tutors attempt to explain these principles a predictable dull glaze clouds most students and learning ceases. The quality of effort of a human tutor teaching boring basics deteriorates with repetition, so the computer is noticeably superior.

A successful recipe - Adelaide Biochemistry

Ten Olivetti M24 computers (IBM-PC compatible) have proved adequate to provide CAI for 300 undergraduates. Each unit has a colour screen and a 20Mb hard disk with courseware selected from a boot-up menu. Questions are available throughout each term relating to lecture coursework. The motivation to take the sessions derives from

- relevance (the questions may be asked in an exam)
- local authoring (Course lecturers are responsible for question generation or selection)
- objective self-assessment of progress.

Students can make their own time to use the CAI materials. This can be at times that are uncommitted or quiet time in a laboratory period.

The local generation of material is important, in part because it shows students that their teachers have made time to prepare the material. The exclusive use of third party courseware is courting disaster. The "Q" Instruction Package has been the basis for rapid generation of local material at Adelaide. The "Q" Package makes session authoring essentially an exercise in text preparation.

Towards ultimate teaching technology with videodisc

Colour enlivens a computer screen because it can impart additional information such as separating factual information from question and answer segments of a text display. Graphics too can contribute to enlivening computer tutorials. It is in the area of graphics and animation however that the most atrocious misuses of CAI authors' time are committed. Figure-1 illustrates a graphic from the pH tutorial mentioned before which is adequate for the purpose intended and took relatively little time to draw with a mouse assisted graphics package. Graphics are in general a patent failure of computer technology to adequately represent real objects statically or in animation. This is the role for videodiscs.

Videodiscs contain a library of 55,000 static pictures in high resolution which can include animated segments (like a video tape). The feature that sets videodisc apart from video tape is that each frame of information can be rapidly called under the control of a computer. Voila! you have it - the ultimate educational technology of the present day. The interactive intelligence of computers linked to high quality pictures stored on a videodisc.

At Adelaide University we were fortunate to have accumulated a substantial resource of slides and film/video which we were able to transfer to video disk in 1986. This has been linked to the "Q" Instruction package for teaching in Obstetrics and Gynaecology and in Geology (figure 2).

An application for teaching written-Chinese at Adelaide required a purpose written program. This application exhibits many of the good points of successful CAI with interactive videodisc and will be discussed.

Conclusion

A successful CAI facility which is likely to survive pressures of obsolescence can be built from current microcomputers. The software must be clearly relevant to the course of instruction. Simplistic computer graphics are generally unsatisfactory, and interactive videodisc is the solution to interactive instruction involving detailed pictorials.

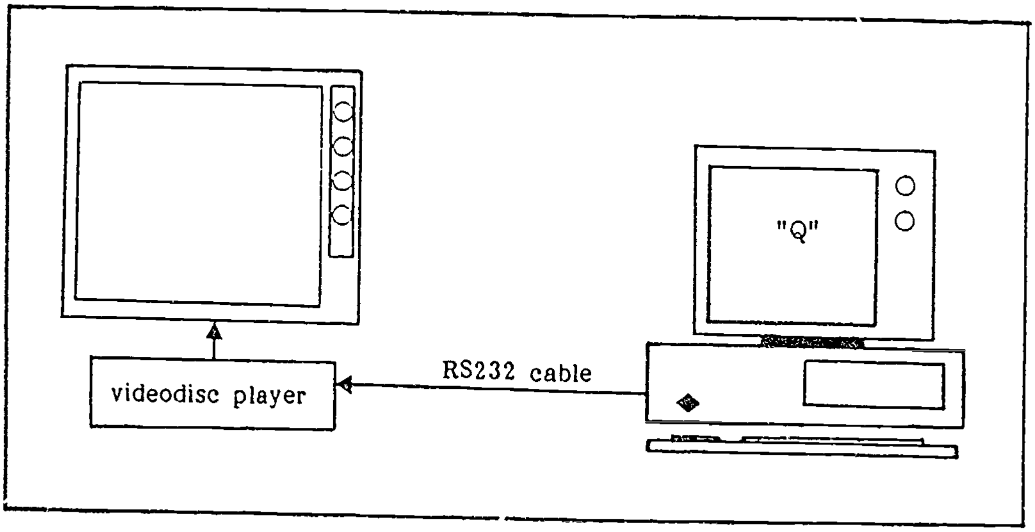


Figure 2: Using the "Q" Instruction Package with videodisc is a rapid means of authoring.

Intelligent CAL, or intelligent CAL users?

Rodney Staples
School of Electrotechnology
Royal Melbourne Institute of Technology

For some time now, the development of CAL has been focussed on hardware, software, and courseware. This paper shifts the attention momentarily to the users. Regardless of the sophistication of the technology, the long term success or failure of CAL depends on it being intelligently applied. Indeed people must choose whether the technology should be applied, and, if so, how and when. Just as the "intelligence" built into many new CAL systems does not happen by itself, people must be adequately informed and prepared if they are to exercise their judgment intelligently.

The paper considers how this human resource might be developed, since the human "intelligent user" is better placed to seek new and innovative applications than is any currently extant computer program.

Since the study of computer assisted learning (CAL) began, research effort has centred on hardware, software, and courseware. Much of this research is derived from work in programmed learning, and training. More recently, systems for learning are derived from the study of artificial intelligence (AI).

So far, computers have not been programmed to be particularly creative. They cannot yet make the quantum leap of understanding which leads to a new use for an existing tool. They cannot invent new tools to solve existing problems. Human users can.

Thus more is needed than inanimate, even intelligent, technology if CAL is to be nurtured and grow as a discipline. Intelligent users are at least as important as the technology. It is the users who will ultimately determine a project's success - or failure.

The skill to use this new technology fluently does not come naturally or easily. It must be developed, and inspired. Once acquired in theory it must be practiced.

People are Important

The importance of people in the development of CAL is shown clearly in the literature. A few examples, by no means isolated, may illustrate this.

In an international study of factors influencing CAL development Hawkins (1977) reported that "In all countries the overwhelming response was the importance of the initiative taken by one, or at most two to three project members. ... Most frequently commented on was the importance of an energetic, dedicated and knowledgeable central figure who could give leadership over the range of activities, and an enthusiastic, cooperative group prepared to put in a strenuous work day over a long period of time."

Sprecher and Chambers (1980) studied 200 CAL development sites. They observed: "They heavy development group was composed of faculty who placed a high value on CAI [Computer Aided Instruction] and who perceived their department, their campus computer center, and the top administration of the campus as favouring it."

The importance of giving users flexibility to adapt the system to their needs was part of the philosophy behind the development of Computer Managed Learning (CML) at the Southern Alberta Institute of Technology in the early 1980's. Stephen (Distefano & Stephen, 1982), commented: "The strength of the system [is] that it lets users be creative, and new applications and methods of use appear as users adapt the system to their particular needs." This implies that the system, which has been successfully marketed throughout the world, was specifically designed to acknowledged that people must still exercise judgement on the use of the system.

Ausburn (1982), reporting on a study tour of the US in which she investigated the training approach adopted by the US Air Force, also identified understanding and commitment displayed by administration and users as significant. She highlighted training as vital in establishing both understanding and commitment.

These examples serve to illustrate that committed and informed leadership is necessary for CAL to be made to work.

They also suggest that developers and users must be informed if they are to be innovative in their use of CAL. Training of some sort plays an important part in the development of the people who will implement CAL. They are all positive examples.

But there are also factors at work which contribute to resistance to the use of CAL. Zemke (1984) found that: "... one distinguishing characteristic of the 'computerphobe' cannot be explained away by our sample: He is absolutely to be found among the one in five trainers with nary a micro-computer at work in his department. Not only has he virtually no experience with computers, he has never taken a CBT [Computer Based Training] course. In short, the computer reluctants in our midst are most easily distinguishable from the computer advocates by the single characteristic of inexperience."

Clearly then it is important that potential users, not just the committed developers and enthusiastic leaders, must be given the opportunity and encouragement to gain experience with CAL.

But is it enough for the users to be trained or experienced in the intricacies of the hardware, software, or courseware? Probably not!

The influence of the teacher is profound. Clark (1983) suggests that the positive achievement effects for media more or less disappear when the same instructor produces all treatments. Hedberg and Charlesworth (1981) suggested the same phenomenon. They observed "that the success of a media-based system in the class-room depended at least as much on the learning activities organised at the receiving end as on the content that went into the transmitting end." Clearly the teacher who is managing a learning program, and specifying alternative resources to meet technology in determining the effectiveness of the media in producing learning achievement. Thus any training or experience must include consideration of the students who will use the system too.

My own informal observations of students working with tutorial type CAL material suggest that students can quickly become bored or frustrated, frequently not completing the set material, and if given the choice may not return to class. Kearsley and Hillelsohn (1982) support these observations. This happens when the CAL designer didn't fully appreciate the diverse needs of student users. In many applications of CAL this phenomenon is not visible because the students are motivated by some form of compulsion. Certainly then, it requires some understanding of the problems of helping people to work with CAL learning resources if they are to be effective.

Chorover (1984) echos an increasing view that: "It is only in the context of a supportive educational community - a human environment conducive to learning - that the hazards of automation [in education] can be avoided." Inappropriate uses, or uses which lead to boredom or frustration in a learning context will be quickly remedied by a teacher with experience using the medium. The teacher must decide if, or when, the tool should be used. People must dictate the use of the technology, not the technology drive the people.

Even an expert ("intelligent") CAL, which has been specifically designed to capture the specialist expertise of both teachers and subject specialists, needs some user skill if it is to be effectively integrated into the learning environment.

The question which is yet to be resolved is how best to provide this experience.

Developing the Experience

There are four ways in which the necessary experience base can be developed in a lay population:

- Mongolian hordes approach;
- Osmosis;
- Formal courses;
- Strong leadership.

"Mongolian hordes" approach

Solving problems by original development has been mainly achieved by employing large numbers of people to work together at a solution. Coyne suggests the "Mongolian horde" approach, throwing large numbers of people at a problem, will probably be paid for ... with the worst skill shortage ever, and this shortage will be accompanied by escalations of salaries, dissipation of management effort in mitigating the increased staff mobility, and a debilitating feeling of instability pervading the industry." The expectation is that given enough people working in an area expertise will develop and leadership will emerge. This is not always true.

In 1980 the British Columbia (Canada) government contracted with a private company to develop CAL materials for local school divisions. This contract also included strategies for developing the skill of participating teachers. This project sought to involve teachers by offering five hundred dollar incentive grants to teachers who participated in courseware development. It hoped that by generating software in quantity, some quality software would emerge. This grant approach "produced no software of usable quality." In this example, the "Mongolian horde" approach did not work.

A similar scheme was a major focus of the Australian government initiatives in computer education in secondary schools. While this initiative undoubtedly developed the skill of many teachers, it barely scratched the surface of the need, and was discontinued last year after only two years in place. Other initiatives in New Zealand suffered the same fate.* It seems that now the same few faces who have always assumed a leadership role in secondary computing are continuing to do so. Again the "Mongolian horde" approach didn't work too well.

The most successful implementation of the "Mongolian horde" strategy is the concerted effort in England to get two teachers with CAL experience into every secondary school in the country. These teachers were to be the nucleus of experience in the schools from whom others could learn. This program has taken many years to realise its goals, and has been expensive, but has demonstrated that a systematic training program to develop "grass roots" teacher's skills can work. But even given its success, the program has shown that "a lot of the educational software [resulting from the program] is rubbish."

The "Mongolian hordes" strategy requires a great deal of money, and considerable commitment from the administration to support the development program until it can become self sustaining. The English experience shows that, because of the resulting increased awareness of CAL, users eventually became more discerning in their choice of suitable courseware, but it is a brute force strategy which does not systematically strive to develop leaders or breadth of skill among users.

Osmosis

The English experience has shown that if there are enough people in the system with adequate skills in CAL, others will be encouraged to seek the skill too. thus the experience of a few will flow through the system by osmosis. This strategy relies on sufficient skill being available for the experience to be visible and clearly acceptable within the system. Without encouragement from administrators for people to undertake specialist courses, or without rewards for gaining the expertise some other way, there will not be enough committed, experienced people to "seed" the osmosis process or to motivate the computer reluctants.

Formal Courses

Formal courses in CAL methods are the cornerstone of the successful English staff development programs in CAL methods. A major impediment to using formal courses to develop CAL expertise in tertiary institutions is the commitment required. Most academics are, quite rightly, committed to developing skills in their discipline or in administration. They have little time left, and are given little incentive, to develop specialist CAL skills. Thus academics are reluctant to undertake still more training than they already have, especially if it is outside their primary discipline.
Despite this, formal courses still have a place. They can be concentrated and systematic resources for gaining the experience quickly. In tertiary institutions which require teacher training, experiences with CAL could be included as a compulsory part of the course. In the more usual situation where such courses are optional, they still need to be available when users are sufficiently motivated. But the courses in themselves are usually not the motivation for formal trainings.

Because of the reluctance to make use of formal courses, a people development strategy based on them will take a long time to reach most existing teachers. If the staff turnover is small, the time required to get the expertise to where it is needed may be unacceptable.

Strong Leadership

Clearly what is needed to excite teachers about the intelligent use of CAL, is strong, visible, and enthusiastic leadership. This must be two fold.

First those who have made CAL their academic speciality must display their successes to others through conferences like this one, in seminars, or even over coffee or lunch. These same experts must be able to discuss the problems likely to be encountered by novices, and the relative merits of a range of possible solutions. Using this already established human resource in the institution as the seed from which the osmosis process can spread may mean that it is not necessary to present courses just to generate the initial experience. Rather, such visible examples might encourage the CAL reluctants to sample the formal courses and explore other learning resources for their own development.

Second, administrations should recognise that the people who can provide the leadership already exist in their organisations. This resource must be nurtured. Administrators need to provide the incentives, both to keep the experts interested in pursuing their new discipline, and to provide the incentive for novices to emulate their more experienced colleagues. For this reason, successful work with CAL should be recognised as valid work for promotion and other rewards.

The advantage for administrations in adopting a positive attitude to building on the existing commitment is considerable. It is not then necessary to generate a sea of novices in the hope that strong leadership, or even adequate experience, will emerge. The resource is already there. The commitment is already there. the ability to make the quantum leap leading to new and innovating uses for the technology is in place. All that is needed is the support to maintain and encourage the work, and to encourage the spreading of the expertise to newcomers.

Administrative Leadership Implications

In a climate in which economic and political pressure to provide vocational preparation and broad personal development at an increased level is being placed on academia, and no less on industry, there is a strong incentive to look to technology as a savior. There are considerable economic advantages for looking to CAL to increase efficiency. But should this cause us to abrogate our responsibility to people and put our faith and trust entirely in technology? I think not.

We must put at least as much effort into developing the skills of people who will use the technology intelligently. People should be systematically encouraged to seek new and innovative uses for CAL. People are better able to do this than any extant computer program. People are the key to successful development and use of CAL.

The incentive to develop the necessary skills must come from the institutions if there is to be a significant development of existing staff to effectively apply CAL. For this skill base to be developed systematically, strong leadership is essential. This leadership must be accompanied by users of CAL who are encouraged by example to improve their own skills. It is these users who will eventually make CAL work. Administrations cannot expect that adopting a "Mongolian horde" approach, or expecting the skills to develop by osmosis, will be effective without strong, effective leadership, backed up by formal training where necessary.

Is it possible that, despite the intelligence built into the technological tools, the real intelligence in CAL needs to be first in the minds and hearts of the human users?

References

Ausburn, L.J. (1983). *Report of Overseas Investigation.* Victoria, TAFE Board.

Chorover, S.L. Cautions on computers in education. *Byte,* June, 223-226.

Clark, R.E. (1983). Reconsidering research on learning from media. *Review of Educational Research,* 53, 4, 445-459.

Coll, J. (1984). Cited in *The Age,* 7th August, pp.39, 42 and in a private seminar at RMIT.

Coyne, L. (1987). How the skill shortages can be solved, in Bennett, N. *Computerworld,* p.16.

Distefano, J. and Stephen, A. (1982). Computer managed learning - An overview. Com 3, May, 31-33. CEGV.

Hawkins, A.C. (1977). Computer Based Learning. *A Survey of the Factors Influencing its Initialization and Development.* The Netherlands: Utrecht State University, p.55, 61.

Hedberg, J.G. and Charlesworth, S. (1981). *The discrepancy between the promise and the performance of technology in education.* Inquiry and Action. Papers presented at the Annual Conference of A.A.R.E., 56-61.

Kearsley, G.P. and Hillelsohn, M.J. (1982). Human factors considerations for computer-based training. *Journal of Computer Based Instruction*, 8, 4, 74-84.

Murray, S. (1986). Commonwealth initiatives in computer education and information technology. In T. Salvas (Ed), *Computers in Education: On The Crest Of A Wave.* (pp.64-67). Melbourne, CEGV.

Prokopanko, C. (1982). *Computing in Canadian schools*. Proceedings of the Fourth Annual Conference. CEGV, La Trobe University, 9-17.

Sprecher, J.W. and Chambers, J.A. (1980). Computer assisted instruction: Factors affecting courseware development. *Journal of Computer-Based Instruction*, 7, 2 47-57.

Staples, R. (1986). CAL - *A new era, or an expensive white elephant*. Proceedings of the Fourth Annual Computer Assisted Learning in Tertiary Education Conference. Adelaide, CALITE, pp.272-281.

Tough ultimatum to academia. Editorial, *The Age*, 23 September, 1987, p.13.

Zemke, R. (1984). Training and computers: After the courtship. *Training*, 2, 10, 48-58.

Computer-based training: Networking of autistic and other retarded students in sheltered training establishments

R. A. Coldwell, Information Technology Division
Royal Melbourne Institute of Technology

One of the major problems concerning education of retarded people in Australia is often their geographical isolation. Where they live in urban areas and, consequently, facilities can be provided for them. They can be given specialised training. Increasingly, we are finding that both autistic children and adults are having to be moved to these locations and, often, to institutions away from their parents. The writer argues that, given the use of existing technology, children can now be integrated into the populations of their local schools and their teaching supplemented by use of computer-based learning via communications networks or even by use of modem and telephone lines. This, in turn, enables them to become integrated into TAFE and other courses and, later, into sheltered workshops which are computer-based.

This is a working paper which reports some of the difficulties with trying to break-down a few of the barriers that we set-up against using technology to bring education to those who, for some reason, are not in a position to claim it for themselves (see Coldwell 1984). This may seem like a curious statement to make in modern-day Australia but, nevertheless, it is valid. Often we set-up barriers in our minds - inhibitions which fashion how we classify people and, consequently, how we treat them. This can be a staggeringly rational process. One influence on the form of education, that intellectually handicapped people receiv , ' the workings of government bureaucracy. Government policy suggests, for example, that disabled children should be integrated into ordinary schools.

Integration into normal schools

The "integration policy" - as it is now called by both parents and the newspapers alike - probably emanates from an administrative need to economise, on behalf of taxpayers, regarding the cost of special schools to the community at large. However, the argument for integration rests mainly in an egalitarian myth which suggests that handicapped children should have the same form of education as other children. This is supported by a behavourist argument that handicapped children will, in mixing with other children, become as normal as they are likely to. This argument is supported by the parents of handicapped children but refuted by those of others who believe that, by interacting socially with handicapped children, their children, too, will become handicapped. These arguments, given by parents and others on both sides, are simplistic. There is, however, some behavioural validity to both sides of the argument.

Integration at the classroom level

What this paper addresses is some of the possibilities regarding using computer aided learning as a mechanism for integration (see Coldwell 1983). Before discussing this, however, we should look briefly at some of the difficulties that occur, at present, in integrating handicapped children into ordinary classes.

Normally, a teacher in an ordinary school can manage, say, thirty children for a while, at least, before, due to tiredness, anarchy occurs. In a special school, a far lower teacher to child ratio - of, maybe, one teacher to five handicapped students - is considered normal. Integrating one handicapped child into a normal class of, say, twenty children can cause pandemonium. A teacher, if she is able to manage, reaches exhaustion quicker and, clearly, needs the help of an aide to manage the handicapped children. This refutes the false argument concerning funding economies. Chaos still occurs. But the handicapped children are being managed, often, by untrained aides and the concentration of the other children is still disrupted. This would, one assumes, affect the progress of all of the children both handicapped and unhandicapped.

Some brief thoughts on autism

It is necessary to explain what kind of disruption occurs. To do so, I will try to describe one of the worst instances of communication disorder that there is. This is called "autism".

Autistic children are often mute, withdrawn, hyperactive, and uncommunicative and, therefore, unable to read or write because they have difficulty in attributing meanings to alphabetical, numerical, spoken, sound and body language symbols (see Coldwell 1985a). Their response to social interaction often appears to be non-existent. Seemingly through frustration, some develop rages. Often their behaviour is interpreted as anti-social whereas it is, we believe, merely asocial. Often their actions are repetitive and, perhaps, ritualised. They may do the same thing continually for hours seemingly with no reason. The target of their fascination is, sometimes, a washing machine, a door handle or a similar piece of equipment (see Montessori teaching theory).

A simple instance will demonstrate one major difficulty with teaching autistic children. If we imagine that an autistic child could ask a question, the following dialogue might occur :

Child:	"What's your name?"
Me:	"Roger!"
Child:	"What's your name?"
Me:	"It's Roger."
Child:	"What's your name?"
Me:	"I said 'It's Roger!'"
Child:	"What's your name?"
Me (now in frustration):	"My name is Roger Coldwell."

If we examine my responses, we see that I have given four different replies to the same question from the child. To an autistic child, this is fickle; hence their confusion with normal social interaction.

Their communication ability seems to be able to cater with logic but not with the illogical component of (and certainly not the humour in) our communication. Let us imagine a similar communication with a computer :

Child:	"What's your name?"
Computer:	"Fred!"
Child:	"What's your name?"
Computer:	"Fred!"
Child:	"What's your name?"
Computer:	"Fred!"
etc., etc.	

This constant play on logic seems to be necessary for an autistic child to see a pattern in the learning process. It is found in the user friendliness of the response of a computer of enquiries to it. Computer aided learning for autistic people is, therefore, strangely successful although it has yet to be demonstrated to the satisfaction of the psychological world which has, in general, taken a hostile standpoint towards its use as a learning mechanism for autistic children.

Computer aided learning for autisic children

For the last thirteen years, the writer has been working with autistic, Down's syndrome, epileptic and a few other types of intellectually handicapped children (see Coldwell 1985b, 1987b). This has involved him experimenting originally with highly structured computer aided drafting (CAD) systems (e.g. RUCAPS) and, later, simple educational graphics-based systems (e.g., LOGO) (see Coldwell 1986). The former were over-structured and the later are too primitive to be of practical use as a computer aided learning tool. A reasonable compromise is the new range of microcomputer based CAD systems like "Autocad". The benefit of using this sort of software is its logical though flexible structure (see Coldwell 1987a). (These have now, of course, been surpassed by total systems, like Plato, which are available on our communications network.)

I normally work with a child with a dual control so that the child uses a mouse while I communicate with the child via a keyboard through the microcomputer. The process that I follow involves the child developing a set of their own drawings through a stage involving hieroglyphic meanings into use of alphabetical symbols. Hence, they draw a cat, store it as "c" for "cat", retrieve it and use it to create pictures together with "dog", "house", "mummy" etc. Eventually, I save the symbol with the full name of "cat" needing them to extend their vocabulary to gain its use. Initially, retrieving three "cats" will involve them pressing the "c" key three times but, eventually, they will use the number "3" to do so etc. The process involves their ascribin meanings to the symbols. Using a pre-se library of symbols would not do this of course.

This whole process has severe limitations however. Normally I travel to them or they travel to me with a parent. This is an incredibly time consuming process for both sides and involves expensive travel for people who already have a large part of their single-person income eaten away by travel costs and other related expenses. (Appreciate, here, that the mother's time is often spent totally on supervising the autistic child.)

Networking schools

Our first move to counter the geographic problems involved in communicating with schools is to use Plato on the Vicnet Communications Network to communicate, from the Royal Melbourne Institute of Technology, with the Warrnambool Institute of Advanced Education. At the Warrnambool Institute, there are Departments of Teaching and Computing and an external Studies Unit which is, at present, giving distance courses to paraplegics. (The Department of Teaching already has a course in special education). This is an initial phase which will precede extending the network out to a school in the Warrnambool area to enable a teaching aide to connect-up with the Warrnambool Institute as a Centre for western Victoria. This will enable direct contact between RMIT and children in classrooms in western Victoria or, alternatively, enable similar contact between them and someone at the Warrnambool Institute. Where parents of handicapped children have microcomputers at home, they, too, may be able to connect-up to the Vicnet communications network.

Centre for social computing applications (CeSCA)

Computer aided learning focal points are needed in tertiary institutes throughout Australia to create centres for social computing applications. Networking schools to integrate handicapped children is but one of their possible functions. Another is the creation of software that is needed to assist teachers to communicate, in dialogue with children in one-to-one interaction, while much of the repetitive learning is managed by child-machine communication.

What have we done regarding this? Throughout the last couple of months, I have advertised in Departments of Computing of the tertiary institutes throughout both Victoria and Tasmania (with astounding success) asking people to write software for intellectually handicapped children. The advertisement has also been reproduced in the "Bulletin" of the Australian Computer Society; it has appeared on noticeboards of computer rooms of commercially orientated organisations as far away as Adelaide and been reproduced in most of the newsletters of Associations for people with different types of disability in Victoria. (As the writer was working, a moment ago, the phone rang with a call from Brisbane.) I have also appealed to our own faculties for obsolete microcomputer hardware and received a mixture of different machines that we will be using to create software for the use of intellectually handicapped and disabled children who are being integrated into schools.

The function of this installation could also be as a computer laboratory for the use of disabled students of the Institute ("Dislab") who are disadvantaged, compared with other students, because of their disability. It can also be used to coach members of staff who are apprehensive about entering their own computer laboratories as novices.

Most important of all of our plans is to use communications networks to communicate with children in schools throughout Australia to familiarise them with the Institute. In communicating with the RMIT, via modem, they will be one step closer to tertiary education. We hope, too, that many disabled students will be amongst them. Some part of industry will, we hope, develop sheltered workshops to enable our disabled graduates to work, alongside their more able colleagues, in industry.

References

Coldwell, R.A. (1983), *Computerised Learning - a sociological perspective.* Proceedings of a Conference on computer-aided learning in tertiary education, August, University of Queensland.

Coldwell, R.A. (1984), *Prerequisites for computer-use - some analogous theoretical systems.* Proceedings of the 6th Annual Computer Education group of Victoria Conference. May, La Trobe University, Melbourne.

Coldwell, R.A. (1985a), *You don't teach adults computing in the way that you teach children.* Proceedings of an Australian Computer Society Conference, Phillip Island, and later published in the Australian Computer Society Bulletin, April, pp.15-16.

Coldwell, R.A. (1985b), *Autism, Montessori teaching theory, computer aided learning and autistic communities.* Proceedings of the Calite '85 Conference on Computer Aided Learning, University of Melbourne.

Coldwell, R.A. (1986), *The role of computers in assisting autistic children : some case studies.* Proceedings of the IBM Inaugural Conference on Educational Trends, April, Melbourne.

Coldwell, R.A. (1987a), *Computer aided graphics and retarded children.* Proceedings of the Second National Autocad Conference, March, Melbourne.

Coldwell, R.A. (1987b), *Computing and disability.* Invited to read paper by the Australian Computer Society at the Sciences Club, Melbourne, which was later published in the Australian Computer Society Bulletin, August, pp.4-5.

The author is a sociologist and a temporary lecturer with the Information Technology Division of the Royal Melbourne Institute of Technology in Melbourne. He is also a Research Associate of the University of Melbourne where his post-doctoral research has been funded by the Department of Community Services Victoria and the Sidney Myer Fund of the Myer Foundation. Recently he has received funding, too, at the RMIT from the Apple Education Foundation. He plans to liaise with the Lincoln Institute of Health Sciences, which is being merged with LaTrobe University, to develop his work further in this field.

Decentralized learning on the tertiary campus: CAI to increase student music literacy skills

Vanda Weidenbach
Nepean College of Advanced Education

Traditional approaches to teaching Music in both schools and tertiary institutions have not led to universal Music Literacy. Graduate teachers on entering primary service, aware of their inadequate music skills, generally avoid teaching the subject. As a consequence, the status is perpetuated. To redress this situation, there is an urgent need to institute innovations, at the tertiary level, which will ensure the acquisition of adequate music teaching skills of a level sufficient for the primary classroom, especially since the introduction of the 1984 Music K-5 Curriculum which makes the teaching of the subject mandatory.

Organizational structure of many teaching programs does not permit the learning of music skills to be evenly spread over time so frequently what is learned in isolated blocks is not maintained. In addition the very nature of music practical skills, in particular, demands time for consolidation and maturation of coordinated movements.

It is proposed that if students take one hour units spread over their entire program using Computer Managed Instruction to supplement formal block semester lectures, [illegible] eral benefits may result. It is accepted generally that students learn at different rates. Consequently, when individual control over how often to take instruction is made available, and when programs that only permit progress after mastery levels have been achieved are used, resulting skills should be of a higher order. Furthermore, because learning has been spread over time, these skills are more likely to be maintained.

As a trial for this hypothesis, the writer has begun a study with a group of first year students at Nepean C.A.E. for whom a controlled number of trials on a specific program will be conducted. Student progress will be individually monitored and ongoing changes emanating from student data will be made. A program designed by the writer for primary school children will be used for the initial trial. One accepts that many of our students have limited computer skills at this level and the program has been designed for the novice. After this first study, it is anticipated that, depending on student response, the program will be modified for the mature student. The writer is also currently investigating the status of C.A.I. in other teacher training programs within Australia.

Music education in the tertiary sector

One of the major concerns of music educators in the tertiary sector is to determine the most efficient way to make use of the limited time allocated to the teaching of music in the preparation of teachers for primary service. The majority of students entering pre-service primary training exhibit limited musical skills in terms of music literacy and practical musicianship. Since 1984, the release of the N.S.W. Department of Education Music K-6 document has made the teaching of music mandatory, so there is an urgent need to ensure students graduate with adequate skills to implement the curriculum. In the past, the teaching of music has proven resistant to traditional approaches.

The proposition that all individuals have the potential to develop music skills to a level which enables them to participate in understanding basic concepts and expressing musical ideas through creative endeavours is indirectly supported by the Music K-6 document. Those of us committed to music education have long held the view that innate musical ability is not essential to achieving a level of skills commensurate with the level demanded in any other subject area. For example, all children receive education in the language arts which will ensure a facility to communicate throughout their lives and there is no expectation that this learning will necessarily result in literary skills of an exceptionally high order. The same proposition holds true for the learning of music skills and therefore one should not expect that children receiving a music education will necessarily become professional musicians.

For those whom psychologists would identify as musically "gifted", a general music education in the fundamentals of literacy, creativity, performance and auditory imagination and analysis, will provide a sound foundation. For these individuals, studies in music of a more intensive and rigorous nature may result in the acquisition of high order music skills leading to either committed amateur or professional musical status. But our major thrust should be to ensure a fundamental grasp of music literacy skills for all children. Therefore, just as language studies are taught, in order that individuals may communicate in a social environment, so should music skills, of a level adequate to enable children to engage in a variety of music activities, thoughts and discussions, be learned.

It is acknowledged that as adults, we do engage in such processes. The recent Report on the Music Industry by Dr. Hans Gulberger (October, 1987) has highlighted the extent to which Australians are involved in music. His data show it to be a 1.6 billion dollar industry employing more than 60,000 individuals, as well as a growing industry, with the numbers of music teachers and musicians having doubled since 1961. This increase seems to parallel the growth of music through the mass media and the exposure society now has to various forms of music which includes popular, jazz, classical European and to some degree classical non-European idioms, as well as folk music. The degree to which individuals are involved in music varies and the level at which one may be involved, diverse, but the size of the industry suggests that the greater majority of us have some involvement which further reinforces the proposal for universal music education.

In the past, the "musically gifted" have been the main recipients of individualized music instruction, frequently receiving tuition from the private sector. Less attention has been paid to the educational mainstream. The main reasons for this have been firstly, the lack of sufficient music specialists and secondly, the generally limited skills of the regular classroom teacher. From a financial point of view, the proposition of specialist music teachers for all schools is unpalatable to the funding bodies. The acquisition of high order music skills in the areas of instrumental and vocal performance skills, composition, analysis, aural skills and the like, which form the basis of a music specialist's expertise, take years to attain. The cost therefore to produce sufficient music specialists for the primary classroom would be considerable.

The alternative, which has been promoted as having a sounder philosophical basis, is to train primary teachers with a level of music skills which will enable them to understand and manipulate musical concepts sufficient to satisfy the demands of the curriculum. (As a side issue, it is an unfortunate fact that accountability of teacher training courses in this regard is non-existent. I am not aware of any analysis which has tried to correlate the skills of primary teachers when the exit their courses with the demands of the classroom, taking into account the existing primary curriculum.) Nevertheless, this should be the aim of preservice primary programs.

And here we find another difficulty which is related to the general structure of teacher education programs whose units, for convenience, are in large blocks. For example, in my own institution, First year students have a one semester Introductory Course in Music at the beginning of their training, but there is a period of two semesters before they meet the subject again by which time the skills achieved in the earlier encounter are greatly diminished. Music is a subject area which demands continuity, time to consolidate and extend both the intellectual processing of musical concepts and practical motor skills for instrumental development. Two final considerations need to be addressed; rate of of student learning and the entry skills. Taking the first, educators accept that students do learn at different rates and we know from the extensive studies undertaken in special education that even the severely and profoundly handicapped will acquire skills, given appropriate teaching procedures and programs. For these individuals, *intensive* teaching *over time* will result in the acquisition of skills not so long ago thought of as unattainable in such a population.

If we apply the same reasoning to the acquisition of music skills at the tertiary level, it is feasible that trainee teachers will attain a level of music skills which will satisfy the needs of the clasaroom, if their learning can be *spread over time* and providing *intensive training* is available. Until recently these conditions could not be met. However, if students are given access to an interactive teaching medium, available throughout their program, and for as often as is required to reach a mastery level, we see a more promising scenario.

Looking at the second condition, that of the very diverse skills levels of students entering pre-service primary education programs, we find that Introductory Studies in Music Education is not only boring for students who may have taken music at year twelve level, but also a waste of student time which might well be used in either extending their existing skills in music or remediating deficits in other areas. At the other end of the spectrum, there are those students who do not possess the most rudimentary music understanding and for whom additional teaching is essential. If these students have access to an interactive teaching medium, on an individualized schedule, they will all benefit in terms of time spent to reach a predetermined criterion.

So, in summarizing this preamble we conclude the following;

- music plays an important part in adult society
- all children have the potential to acquire music skills
- all primary teachers need basic classroom music skills as prescribed in the curriculum
- music learning during preservice training needs to be spread ove. time
- students learn at different rates so access to learning according to need will be more efficient and effective than group instruction
- both C.A.I. and C.M.I. have the potential to provide such access

We therefore hypothesize that;

- decentralized learning on the tertiary campus, used in

- conjunction with more traditional approaches, should lead to increased music literacy for trainee teachers and ultimately, for all primary school children.

Computers as music teachers

At the very outset we wish to emphasize that Computer Assisted Instruction in music is not expected to solve all the problems of Music Education in Teacher Training. However, clearly it does have the potential to assist tertiary music educators in overcoming some of the existing difficulties. No-one would suggest that all music courses can be taught more appropriately, or even at all, using the new medium any more than one would suggest that any one method of teaching is justifiable. An eclectic teaching style which ranges from direct instruction procedures to exploratory learning experiences is eminently suitable for the varying components of music learning. There are those factual aspects of music which remain unchanged such as pitch, intervals, history, analysis and the like while other endeavours such as composition or an understanding of orchestration, are eminently suited to a discovery learning approach. (How many compositions or orchestral arrangements anguished over by students during training, never reach their ears through instrumental realization?)

Synthesized sound capabilities of the microchip will make the acquisition of these skills in particular more efficient using C.A.I. as students manipulate their attempts while simultaneously receiving aural feedback.

The microcomputer is also eclectic in the various teaching procedures it can present, again eminently suitable for the learning of a wide range of music skills. Its ability to present text and musical notation through tutorials for skills acquisition, to provide drill and practice for speed and fluency, to provide simulations, games and problem solving activities for generalization and extension, make the microcomputer a versatile instructional medium. Its facility to allow interaction between teacher and learner, further enhance its qualities as an educational medium.

Many advantages, well documented for other subject areas are relevant to the teaching of music and, in addition, there are special features which are particularly appropriate to music learning.

In general terms, the computer is able to;
- pre-test students so they enter the program at an appropriate level
- present music sounds, notation and written text
- analyze student responses, both written and musical
- provide objective, instantaneous feedback
- as a consequence, increase potential learning time
- give immediate and appropriate reinforcement
- keep student records and identify, for the lecturer, those students in need of additional help
- enable students to be actively involved in the teaching/iearning process
- provide motivation through moving graphics and sound
- provide further motivation because of the confidentiality of learning
- spread learning over time
- give students control over the amount of learning required for mastery
- block students from progressing until mastery is achieved
- give student access to learning outside the lecture room
- move students to new levels or offer remediation, as required.
- provide students with programs which will ensure a level of basic skills prerequisite to their courses, perhaps in the

form of a bridging unit.
- permit lecturers to proceed at a faster rate because of acquisition of these skills thereby leading to further extension of skills

In more specifically music oriented tasks, the unique characteristics of the computer offer;
- the provision of moving graphics to illustrate certain concepts of music which is particularly relevant, given the transient nature of sound
- the provision of sound itself, now available with most microcomputers
- the ability to present monophonic and polyphonic sound
- the attachment of the MIDI (Musical Instrument Digital Interface) through which certain synthesizers, keyboards, sequencers and samplers may be connected to the microcomputer
- the ability to reproduce at an ever increasingly sophisticated level, the sounds of conventional instruments as well as previously unknown timbres
- the printing of musical notation and scores
- the ability to analyze information and keep data bases'
- the ability to receive sound stimuli from the student, analyse it and report to the student

There are also direct benefits for the teacher because the computer allows him to;
1. pre-program systematic instruction
2. ensure relevant sequencing
3. revise and update the program according to student performance
4. individualize aspects such as reinforcement schedules according to student needs
5. be relieved of repetitious teaching
6. keep records automatically
7. spend more time on class management and organization of programs

Not only is the computer able to provide direct teaching, it can also provide a means of allowing explorations into the domain of creativity. "The clearest trend in C.A.I., " (Upitis, 1983) " is that computers are being regarded less as a vehicle for delivering programmed instruction and more as a means for allowing children to actively manipulate variables and solve problems."

Generally, music educators would agree that the most efficient way for children to learn music is for them to experience it in the most practical ways whether through movement activities, singing, instrumental experimenting, or creating their own compositions. Seymour Papert's theories concerning the acquisition of mathematic skills similarly suggest that children should "do" maths rather than "know" maths and through LOGO this has been made possible. The little known LOGO music system, was developed primarily to teach composition.

Logo is a computer language written in the computer language Lisp which has been much used in programming artificial intelligence. (Baugh, 1987). Despite this, its principal function is " not so much to allow children to engage in computer programming but rather to act as an environment in which they may learn about the particular subject matter in hand and also to learn about their own thinking." (Stevens, 1987) Thus in terms of problem solving activities, Logo encourages the child to explore his critical thinking skills. The two primary types of Logo are Terrapin Logo, developed at M.I.T. and the other, L.C.S.I. Logo from Logo Computer Systems. The former includes Commodore Logo, P.C. Logo for the I.B.M. P.C. and Krell Logo and Terrapin Logo for the Apple ll series, while the latter is used for Apple Logo, Apple Logo ll and Sprite Logo for the Apple ll, Atari Logo, IBM Logo and Logo for the Apple Macintosh. (Baugh, 1987) Perhaps the main disadvantage of Logo is that one cannot use traditional notation or note names to define pitch or duration with numbers or letters being used to assign these elements. For more detailed descriptions of the use of Logo music, see Stevens, (1987).

There is now a number of high level languages which can be used by teachers to create their own programs. Super-PILOT allows one to program sound, graphics and text relatively easily using a series of mnemonic, alphabetic commands to control the operation of programs. Music teachers' entry into this part of the process "may result in a fresher approach to some teaching problems." (Placek and Jones, 1986). It will certainly encourage music teachers to be analytical in lesson planning. Clearly, music c.a.i. is more than drill and practice of theory as so many music teachers believe.

Continuing research on the effects of using computers to teach music is an important issue, not only informative on the specific outcomes of studies but also for the general conclusions it produces. Placek's study (1974) with students majoring in elementary education presented a C.A.I. lesson on rhythm, recording data on total time spent, amount of program covered, amount of time on each main routine, number of attempts and corrects and errors, and other aspects. The post intervention data with students found they enjoyed the lesson and learned the basis behaviours taught, and as well, they identified some deficiencies in the program. However an even more important outcome related to the development of goals and objectives for music education generally.

> "Because of the deterministic character of digital computers, identifying and defining musical behaviours seems to be a natural outcome of planning, programming, and producing music lessons for computer aided education systems."

In the past, many music teachers have been reluctant to state objectives on the grounds that music is an art rather than a science. However, the demands of programmers in developing software may lead to improvement in this area as specifying precise learning outcomes is an essential part of instructional programming.

One further aspect not previously considered is that of student motivation. Certain computer programs have been found to be highly motivating, an important aspect of all learning. Recognizing the powerful motivation factor in video games, Chaffin et al (1982) called for the identification of those variables associated with the motivational aspects of the games so they could be applied to educational settings. Malone's study (1981) resulted in the general classification of motivational aspects under three categories- challenge, fantasy and curiosity. Fredericksen et al (1982) described three computer games designed to improve reading skills which incorporated the motivational features of video games. They cited;

1. clear cuts goals,
2. fast pace
3. immediate feedback
4. variable levels of challenge as important features.

In designing software for music, consideration should be given to this issue.

Should we as music teachers use computers to teach our precious art? Is there justification for bringing technology into the music room? Can the computer teach as well as the teacher? Can it possibly teach some aspects in a superior way?

One crucial aspect of music learning is that of repetition. The skill intensiveness of the subject whether it be for developing the motor skill of instrumental playing or the more intellectual processing of recognizing and analyzing sound patterns, demands a high level of individualized instruction and repeated practice which in the past has been difficult to provide. Without the direct intervention of a teacher the microcomputer can:

- present a stimulus
- accept a response
- evaluate the response
- give feedback
- give immediate and appropriate reinforcement
- move to the next instructional sequence or branch to further explanations
- record results

And repeat the task endlessly.

According to Hofstetter (1979) "there is no other discipline for which microelectronics are better suited than music education. Although still in the infancy stage, microcomputers are able to produce music notation, generate aural stimuli, judge responses, and diagnose student learning." Given appropriate software based on sound instructional design and with appropriate teaching procedures, many aspects of music knowledge can be learned with minimal teacher intervention.

Whether the new technology will be integrated in music education is not in question. "The challenge to education in the 1980s is how best to use computer courseware to improve instruction." (Wilson and Fox, 1982).

If music education is to survive, it must accept this challenge. Not a panacea for the problems of music education, the computer, because of its interactive nature, and the versatility of teaching procedures, is a medium unlike any other teaching aid, and holds great promise.

Peripherals to enhance performance in music learning

In addition to the microcomputer unit itself which includes an attached qwerty keyboard to access information, there now exists a number of attachments known as peripheral devices which enable musicians and music teachers to more easily access the computer. During the 1960s and 1970s, music specific attachments were developed which were expensive and sometimes cumbersome. Since 1983, the development of MIDI,(Musical Instrument Digital Interface), has revolutionized communication between computer and various musical instruments. With suitable software, a synthesizer connected to the computer via MIDI can be used as a real-time music input device for composition or performance, for learning keyboard skills or music theory as well as other music activities. Through the music keyboard, notation can be displayed and edited, then played according to the user's directions. A hard copy of the resulting music manuscript can then be printed. In the case of learning piano keyboard skills, the computer is able to direct the student to perform and provide feedback on the accuracy of the student's performance in much the same way a teacher would.

There are various devices which can be connected similarly. These include synthesizers, both monophonic and polyphonic, electronic keyboards, drum machines, samplers, sequencers, and even special sound effects devices, each adding diversity to the computer's regular functions and providing facilities not readily available in the regular music room. Some devices have MIDI inbuilt while those without may be interfaced by an independent MIDI. There also exists a range of synthesizers for non-keyboard musicians which are similar to an auto-harp but having controls similar to those used by woodwind players or guitarists. Expanders, which are sound devices having no direct controls themselves, can be played by another keyboard or synthesizer to extend its tonal range. Mixers, compressors and equalizers may also be used for specific purposes. For further information on these peripherals see N.S.W. Department of Education publication Music and Computers (1987), or Stevens (1985, 1987) and Wells and Kemp (1985).

Music editors for microcomputers, though not originally created for music education, also have a valuable contribution to make. An article by Mercuri(1981) gave a comprehensive description of nine such programs listing some 37 characteristics which provided useful information. "The music editor may be used as a motivating tool for children to pursue additional and more varied musical activities." (Upitis, 1983).

In the past, particularly at primary level, composition and creative musical pursuits have not been high on the list of priorities although the new K-6 Music Curriculum has identified this as an important area. It may have been due to the lack of musical knowledge and experiences of the classroom teacher that this occurred. By the early 1980s it had become apparent that music editors could give children opportunities to experiment with musical ideas and patterns. Having such programs available in the classroom may not only assist the children but also the teachers to gain experience and confidence in such musical activities.

The most recent curriculum documents on music learning both within Australia and overseas, place emphasis on the learning of music through listening, performing and composing. The vast array of electronic musical instruments in their various forms including those which can be interactive with a computer to provide C.A.I., offer to the music teacher and the student ready access to a range of instruments and sounds as well as learning experiences not previously available. It is important therefore that we accept the new technology, capitalizing on its benefits in enabling children to work with another learning tool.

Applications of computer technology in music education

The first reported use of computer assisted instruction in schools was in 1959 in New York State so we have had 30 years to acquire some understanding of the potentials and pitfalls of the technology in general education. Since the late 1960s, there has been recorded a variety of music interventions using the computer as the teaching medium. Much of the earlier instruction was given at universities in the U.S.A. and included such areas as aural training, sight-reading, rhythm perception, analysis and composition.

Allvin's work at Stanford to teach sight singing (1967), Placek's much reported PLATO system at the University of Illinois to teach rhythm tasks, and the work of Lamb and Bates, University of Canterbury in England, which concentrated on aspects of melody, harmony and rhythm were all forerunners to later studies which have grown in diversity. Ned Diehl's work developing peripherals for the IBM 1500 computer to teach articulation, phrasing and rhythm for wind instruments, pointed to new directions in music teaching, (Deihl, 1971).

By 1974, PLATO had produced courses on the following topics; Tests and Measurements in Music Education (Colwell and Weimer); Instrumental Methods Series - wind and percussion (Peters); Percussion Terminology (Fairchild); Violin and Viola Fingering Drill (Lind); Instrument Recognition (Holden); Micro-teaching in Music Education (Dvorak); Critical Incidents in Music Education (Cooksey); Hand Signals for Music Teachers (Flohr); Elementary Music Fundamentals (Placek and Peters); Part Writing (Rickman); and Jazz Chording (Rucinski.) Programs being developed at that time evidence versatility.

One area to receive considerable attention is that of aural training. Such is the intensity required to master the skill, even prior to the advent of the computer, non-teacher centred programs were devised using the tape-recorder. Kuhn at Stanford (1974) and Hofstetter at the University of Delaware (1975) developed computer programs in this domain. Studies by Herrold (1973) and Killan and Lorton (1974) documented the effectiveness of Kuhn's programs. By 1975, with the help of record keeping by the computer, Killam et al, were able to research into how students develop aural skills and how to establish a hierarchy of learning concepts. Hofstetter's work on the GUIDO (Graded Units for Interactive Dictation Operations) system led to research by showing that, compared with a control group, students using GUIDO for ear-training, performed significantly higher. Computerized aural training (Lees, 1980) concluded that "the computer program was in virtually every aspect vastly superior" to her own tape-recorded program. And further, the computer's ability to diagnose individual student's difficulties and present examples of that type, was a particularly significant innovation.

Taylor (1982) using MEDICI, an interactive computer-based dictation system with tertiary students, sought to compare student gains using traditional piano dictation and the PLATO System interfaced with the Gooch Box, a [illegible] music synthesizer. While all student showed gains in skills, those using MEDICI did not produce significantly higher results. However, the computer offers advantages of releasing the teacher for other functions, reduces repetitious teaching and allows students to work according to their own needs and timetable constraints. Woods (1986) in describing a population of young children of whom 30% were unable to sing in tune, stated that less than 3 hours of c.a.i. instruction was sufficient to remediate their problem.

Diehl's students (1971, 1973) using the I.B.M. 1500 showed significant gains in instrumental performance. His studies aimed to have students develop more 'sensitivity and accuracy" in performing. Musical notation displayed on the screen together with pre-recorded musical examples were devised for flute, clarinet, oboe, saxophone, trumpet and baritone. Significant gains especially in phrasing were evidenced. Peters (1975) developed for PLATO a series of programs for flute, clarinet, oboe, saxophone, bassoon, trumpet, horn, trombone, euphonium, tuba and percussion. Using tutorial format, students were automatically moved to more advanced materials or branched to remediation according to their needs.

Programs in fundamentals of music theory, employing tutorials and drill and practice proliferated at the tertiary level in the U.S.A. Studies in Set Theory (Wittlich, 1974); Composition, programs by Nelson and White (in Hofstetter, 1975), Appleton and Alonso (1975); Analysis (Hofstetter, 1972); Whitney (1975); Information Retrieval (John, 1969, 1971); Automated Music Printing (Lelena Smith, 1973) further illustrate the variety of music c.a.i. in the 1970s. Bamberger (1974, 1979) explored the use of LOGO with a group of children who were able to manipulate sounds of different pitch and duration by using letters, number and words. The resulting music is then played by the program.

Through the 1980s there appeared a plethora of courses to teach music fundamentals. Lamb, (1982) developed an interactive system which allows musical concepts to be modeled graphically. His program makes less demands than the LOGO system while still offering similar games using traditional notation. At the same time, work on computer managed instruction began to appear. Recognizing that in individualized performance-based instruction, students do proceed at different rates and experience different forms of learning difficulties, it was acknowledged that, if teachers monitored student progress, they could assist individuals in need. Arenson's development of the PLATO-based C.M.I. led to the conclusion that if students are given self-paced C.A.I. , teachers have more time for individualized instruction on a needs basis.

Arenson (1982) examined the effect of a competency- based education program on the learning of fundamental music skills with a group of tertiary students using the PLATO system at the University of Delaware. His results supported the premise that drill and practice procedures in this area when provided by the computer produced significantly superior results in learning than for the control group.

We conclude that programs to teach a wide range of skills have been successful but how extensive is its use. In a study of 429 music departments in the U.S.A., Jones (1975) concluded the following;

> Despite evidence of the computer's value for instructional uses, the educational community has been reluctant to accept and develop computer assisted instruction for music. Reflections of this reluctance are manifold:
> (1) few music educators are involved in teaching by computer systems,
> (2) few students are involved in research in computer assisted instruction as learners,
> (3) few music educators and graduates are involved in research in computer assisted instruction,
> (4) few quality course materials are available, and
> (5) no formal mechanisms exist for sharing computer assisted instructional efforts in music.

As an outcome of Jones' recommendations, a National Consortium of Computer-Based Musical Instruction was formed in the U.S.A. As worthy as the aims of this body appear, many of Jones allegations still hold true. While articles in a wider variety of journals can now be found discussing music C.A.I., there still exists a dearth of reported research on specific learning outcomes.

Currently in Australia, articles are few and research almost non-existent. The technology is here, the software is becoming more readily available and affordable as well as improving in some aspects, peripheral hardware has reached quite sophisticated levels, justification of its potential value as a teaching medium widely discussed, but development is not widespread.

With the establishment of the PLATO (Programmed Logic for Automatic Teaching Operations) at the University of Western Australia in 1985, using the GUIDO programs there is an expectation that on-going research into the effectiveness of the programs will ensue. In a recent publication (MacPherson,1985), some indication of the extent of the programs to which students can be exposed was given.

> As with any subject requiring the acquisition of literate or intellectual skills, the computer is of most use when providing a graded series of drilled exercises. In traditional aural programmes at the tertiary level, and as reflected on GUIDO ear training lessons, these drill activities are mainly associated with the recognition of intervals, melodies, harmonies and rhythms, and by writing them down via dictation type exercises. The ear training curriculum of the Micro GUIDO System consists of five main lesson types i.e., Intervals, Melodies, Chord Qualities, Harmonies, and Rhythms.

In recognition of the possibility that this type of learning might not suit all students, a two year study has been implemented to evaluate student progress and attitudes. Mr. MacPherson enunciated the problem faced by most music educators at tertiary level, that is the wide range of individual differences in dictation ability, especially at first year level. "When using the computer, students receive individualized attention, with evaluation and feedback of their responses being immediate. Each time a student masters a particular unit in GUIDO the computer automatically writes a star next to the appropriate unit on the student's floppy disc. By releasing the instructor from personal tuition, it permits him to become more of a "diagnostic consultant" who can pay greater attention to specific student needs.

There is an urgent need in Australia to take up the challenge of such research because it is only through empirical evidence that rational conclusions, which will ultimately benefit students, can be made. There exists some evidence of the success of C.A.I. to increase students skills, in overseas studies. For example, the New England Conservatory of Music in the U.S.A. (Lafferty, 1986) reported that students taking c.a.i. in aural training increased their skills by 50% over students who did not use c.a.i. Such promises are too exciting to ignore.

Stevens and Sillcock (1985) began a pilot study with members of the Defence Force School of Music to assess and remediate the level of aural skills of musicians in the Royal Australian Army and Navy bands. To date the results have not been reported but that such studies are beginning at local institutions augurs well for future endeavours in researching c.a.i. outcomes in music learning.

Nepean study

The foregoing evidence led this writer to suppose that c.a.i. in music on the tertiary campus could be used to answer the needs of primary teachers in training and that, in addition to its use as a group teaching mode within the lecture room, individualized instruction of the amount required by each student could assist in raising the levels of music literacy for trainee primary teachers. If units were provided outside the lecture room for which students could gain credit in their own time, such learning could be spread over time thus ensuring maintenance of both practical and theoretical skills.

In 1987, a group of 120 first year students at Nepean C.A.E. undertaking the Bachelor of Education Program in Primary Teaching, were screened for basic music literacy skills. A sample of the students has been selected to use a program designed to teach those skills presented in the pre-test. In 1988, after a predetermined number of trials, the total population will be re-tested and data of the two groups compared. While it would be unwise to predict the outcome, there is sufficient evidence from other research studies to hope that learning outcomes will be significant.

Conclusion

Music educators have long sought means to increase the level of music skills across all sectors of education. If we are to capitalize on the perceived potential of c.a.i., we need data on which to base conclusions, and longitudinal research which can be used for continued development. The new technology has more to offer than any previous teaching tool but it needs the help of music teachers, musicians, curriculum experts and programmers who are sensitive to the importance of instructional design within the framework of the various modes of presentation the computer is able to offer. Without sound, validated design, the computer will not live up to our expectations. Instructional design as an heuristic problem-solving process, depends on intelligent decision making on the basis of incomplete data which must be followed by rapid evaluation and revision according to the outcomes.

Keyboarding for information: Don't let your thoughts get away from your fingers

Uni Carnegie and Michael Gerrard, School of Business
South Australian College of Advanced Education

The rationale for the development of a computer-based keyboard training program for first-year Bachelor of Business students is described. The textual material was developed for use by young adults, but has been used successfully with people whose ages range from 5 to 70 years. A sample of the relevant research into the acquisition of psychomotor skills is provided. Details are given of the construction of the computer program which drives a learning package which has, in its first year of use, provided the means whereby in excess of 150 students have become functionally literate at the computer keyboard.

Micro computers are now being used extensively by undergraduates in our business courses, not so much as a topic to be studied but rather as a versatile tool to be used to support studies in accountancy, finance, administration etc.

Most new students are not fluent in using any type of keyboard, but they do need to develop fluency quickly. Since it is difficult to delete any of the more 'academic' topics from the curriculum, keyboard skills must be acquired largely outside formal staff-student contact time.

These constraints lead to the desirability of students learning with a tutorial program running on the micro computers which they will be using. While many such programs exist for a wide variety of computers, a quick survey of those readily available for MSDDOS machines revealed undesirable features in each case. Some were too tedious, some required an installation procedure for each student, and some were simply too expensive. Hence the authors chose to develop a new program with the following limited aims:

- Use of all fingers at 15 words per minute after 5 hours
- Simple to start and use for students new to computers
- Immediately available at any of 40 micro computers at the College, without the need for a floppy disk
- Easily distributed to external students who use a variety of MS-DOS machines.

One of the authors (Carnegie) has been teaching keyboarding for several decades and had developed a successful method used with typewriters. This method introduces a few new keys at each lesson without any tedious drill exercises; students type complete sentences right from the first lesson. The new computer program simply translates this proven method into a more modern technology.

The first students to learn by this quick method were technical college lecturers learning on manual typewriters so that they could prepare lesson material and handouts to be run off on a spirit duplicator - such was the technology of 1964! They were predominantly men from the Engineering and Building departments. They attended for a total of 5 hours - just one week. By that time they had learnt the whole keyboard, including the numbers (which are often left to last and then forgotten) and gained some basic skills in setting out material.

Since then the method has been used to teach full-time students aiming to become professional typists, as well as people ranging in age from 5 to 70. Some of the people were authors whose flow of ideas was constrained by their inability to get their thoughts on to paper quickly enough. One person came specifically to learn enough to prepare a list of customers and their requirements for Christmas turkeys. How much easier it would be for that turkey farmer now, storing the details on disk from year to year and updating the lists!

Some principles of skills acquisition

Keyboarding is a complex psychomotor skill. It demands the mental co-ordination of muscles to produce meaningful alphanumeric text. The complexity of the process is demonstrated by the factors identified by Fleishman (1966):

- *Control precision.* Highly controlled adjustments of large muscles
- *Multi-limb co-ordinating.* Co-ordinating the movements of more than one limb
- *Response orientation.* Fast visual discrimination with appropriate movement
- *Reaction time.* Speed of response to stimulus
- *Arm movement speed.* Rapidity of gross arm movement
- *Rate control.* Following a moving target
- *Manual dexterity.* Skilled rapid arm movements
- *Finger dexterity.* Controlling tiny objects with the fingers

Fleishman's analysis of the stages of practice on a complex co-ordination task such as keyboarding further emphasises that this is not a simple skill that can be left to chance.

(Illustration - Stages of Practice)

When children learn to ride a bicycle or to surf, they usually study what the experts are doing. They watch and imagine themselves making the same movements. This kind of mental rehearsal has been shown to be useful in establishing neural pathways for the spontaneous execution of precise movements.

When the learners pluck up the courage to launch into action, there is already a mental image of what should happen and they experience kinaesthesis - what it actually feels like. They soon learn the feel of successful movements and may practise for long periods of time to perfect a particular technique as they gradually refine the skill and develop the automaticity which is the hallmark of a skilled performance.

The effectiveness of conscious mental rehearsal in relation to passive observation in shown in the following analysis by Cratty (1973) of several research studies.

(Illustration - Effective pr[illegible]tice)

It has been asserted (Austin & Pargman, 1981) that the most fluent performance is achieved by the person who is able to leave the execution of the skill to the unconscious self. The conscious self attends solely to the higher order activities associated, in our case, with word processing or computer programming.

Clearly, if keying is automatic, with students having no hesitation in locating the required keys, they are going to be able to initiate the actual movement more quickly. This may be of marginal importance in computer programming, but where people are word processing to create original documents, then fluency in keying becomes a critical factor.

Denis Glencross, Associate Professor of Psychology at Flinders University, has carried out a considerable amount of research into the acquisition of typing skills. He has shown that 'hunt and peck' operators show a marked improvement in both speed and accuracy when they retrain as 'touch' operators.

Using the program

The computer program contains 24 standard lessons, each introducing one or more keys, plus three explanatory lessons and two optional lessons. The student chooses the required lesson from the menu in figure 1.

Each standard lesson begins by identifying the new keys and the fingers to be used. The students press each new key, and then type seven sentences which use these keys. Any errors are politely identified for immediate correction, and encouraging comments are displayed if the typing is accurate.

The top half of the screen displays a diagram of the keyboard with the new keys for the current lesson highlighted and the home-keys underlined. This technique encourages the student's eyes away from the keyboard, thus reinforcing the learning of the key positions without the student getting into the habit of 'hunting and pecking'.

Sentences are used from the very first standard lesson, because of the sense of achievement that this gives to students. The sentences are, of necessity, limited in their intellectual content, but the pattern of the finger reaches required enables students to unwittingly establish the correct movements.

Meaningful practice is known to facilitate learning and spaced practice is generally more beneficial than massed practice, as interest and enthusiasm are maintained and fatigue avoided. Nevertheless, it has also been shown that practice sessions must be both regular and frequent.

If, as teachers, we know there are common error patterns, we are bound to try to eliminate them.

Because it was common for students to confuse E with I and D with K, for example, a conscious effort was made in the program to establish one key before introducing the key struck with the same finger on the other hand. E is introduced early in the program because it is the most common letter in the English alphabet and is often miskeyed. I is the key in the same position on the other side of the keyboard. In this program the D-E finger reach is established before the K-I finger reach.

While the program adopts what the Americans call the 'skiparound' method of introducing the keyboard, it was felt that there was value in establishing the baseline of the guide keys (sometimes referred to as the 'home keys'). The value of establishing this base in the first lesson is seen as more important than possible problems associated with, for example, miskeying D for K. The construction of the sentences addresses this issue.

Preliminary results

In 1987 the new program was the first software encountered by 270 business students in the Introduction to Computing unit at the South Australian College of Advanced Education. Their first 75—minute workshop showed them how to switch on the computers, start the program, select lessons, quit from the program and switch off the computer. An average student covered the first 6 or 7 lessons during this workshop.

Students were encouraged to complete all the lessons within the following two weeks in their own time; no further contact time was devoted to keyboard skills. The program was readily accessible for 75 hours each week, stored on the hard disks of 40 micro computers. Although it was up to the individuals to progress through the program at their own rate, they were advised to try to have one or two short sessions (15-30 minutes, 2-4 lessons) each day.

A few weeks later 197 of the students responded to a brief questionnaire. 33% of these had previous keyboarding experience and 36% admitted to spending less than two hours with the program after the initial workshop. The remaining 31% were considered to have made a significant attempt to use the program to gain keyboarding fluency.

No attempt was made to measure their typing speed objectively, but all those using the program believed they could type at least 10 words per minute and almost half felt that they could type more quickly than they could write legibly. Casual observation later in the semester showed that the majority of these students tended to use all (or most) fingers, whereas the majority of the computer programming students in previous years used only one or two fingers.

Although these results are largely anecdotal, they have encouraged the authors to continue with the program in 1988 and monitor the students' performance more objectively. Other institutions or individuals wishing to use the program may obtain it from the authors on payment of a nominal licence charge.

Conclusion

It is because of the overwhelming weight of evidence in favour of 'touch keying' that we require our students to work through the program described in this paper. We do not want to be accused of allowing them to become cripples - people whose thoughts cannot be expressed fully or effectively because they are inhibited by their lack of keyboard fluency.

This computer program appears to meet our requirements of enabling all students to acquire modest keyboard skills without infringing significantly on contact time needed for other topics.

Griffin and Keogh (1981) show that student perception of demands in movement situations may well be the key to understanding students' movement confidence, since individual perception will determine how personally important and useful participation becomes. Movement confidence is particularly important for handicapped students who attempt to achieve effective movement behaviour in spite of movement inefficiencies related to their handicapping conditions.

Are we not culpable, if we allow our students to become functionally handicapped because we do not enable them to perceive the value of touch-keying?

References

Austin, J.S., & Pargman D. (1981). The Inner Game approach to performance and skill acquisition. Motor Skills: *Theory into Practice*, 5(1), 3-12.

Carnegie, U. (1987). *10-Hour Typing Program*. Melbourne: Longman.

Carnegie, U. (1985). *Key in for Information*. Adelaide: Unisco.

Cratty, B.J. (1973). *Teaching Motor Skills*. Englewood Cliffs: Prentice Hall.

Fleishman, E.A. (1966). *Individual Differ-*

ences. In E.A. Bilodeau (Ed.), *Acquisition of Skill* (Chapter 6). New York: Academic Press.

Fleishman, E.A. (1975). Toward a Taxonomy of Human Performance. *American Psychologist*, 30, 1127-1149.

Glencross, D.J., & Bluhm, N.M. (1986). Intensive Computer Keyboard Training Program. *Applied Ergonomics*, 17(3), 191-194.

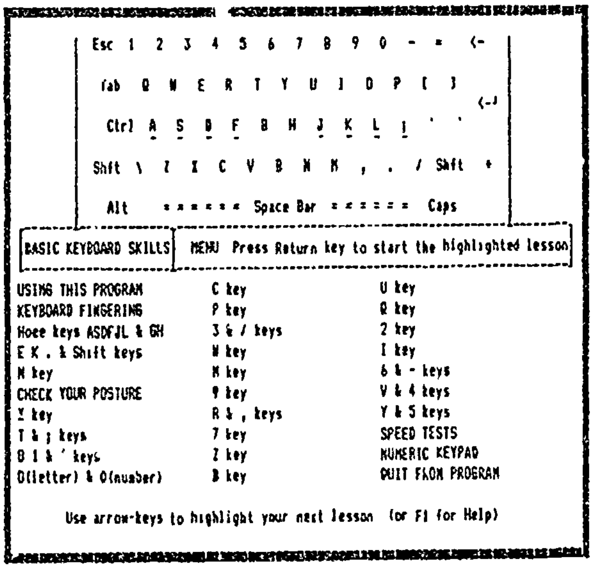
Griffin, N.S., & Keogh, J.F. (1981). Movement Confidence and Effective Movement Behaviour in Adapted Physical Education. Motor Skills: *Theory into Practice*, 5(1), 23-35.

For a detailed psychological investigation of the complexities of keyboarding, including word processing, readers are referred to

Cooper, W.E. (Ed.) (1983). *Cognitive Aspects of Skilled Typewriting*. New York: Springer-Verlag.

Dr Michael Gerrard has worked as a computer programmer and consultant for ten years since completing his doctorate in business management at London University. Now teaching computing to undergraduates in business, he stresses the need for students to develop practical skills by using microcomputers throughout their course. He is a member of the Australian Computer Society and regularly presents short courses for the Australian Society of Accountants.

Uni Carnegie studied motor skills acquisition at Oxford University and has an MA in Education from Lancaster University. She lectures in office administration and is currently supervising a number of student research projects related to office administration. She is a Fellow of the Faculty of Teachers in Commerce (UK) and a member of the Office Automation Association of South Australia, the Business Education Teachers Association of South Australia and the Australian Association for the Education of the Gifted and Talented.



```
   Esc  1  2  3  4  5  6  7  8  9  0  -  =  <-
   Tab  Q  W  E  R  T  Y  U  I  O  P  [  ]
                                              <-'
   Ctrl  A  S  D  F  G  H  J  K  L  ;  '  `
   Shft  \  Z  X  C  V  B  N  M  ,  .  /  Shft  *
   Alt   = = = = = =  Space Bar  = = = = = =  Caps

BASIC KEYBOARD SKILLS   MENU  Press Return key to start the highlighted lesson

USING THIS PROGRAM        C key              U key
KEYBOARD FINGERING        P key              Q key
Home keys ASDFJL & GH     3 & / keys         2 key
E K . & Shift keys        W key              I key
N key                     M key              6 & - keys
CHECK YOUR POSTURE        9 key              V & 4 keys
X key                     R & , keys         Y & 5 keys
T & ; keys                7 key              SPEED TESTS
D 1 & ' keys              Z key              NUMERIC KEYPAD
O(letter) & 0(number)     B key              QUIT FROM PROGRAM

      Use arrow-keys to highlight your next lesson  (or F1 for Help)
```

Figure 1

Development of CBT using AUTHOR

Tony McSherry,
Director, Microcraft Pty Ltd

The history and evolution of the AUTHOR authoring system, design considerations and decisions. The creation of simple lessons and tests is described and demonstrated with particular emphasis on the system's word processor, graphic design system and animation package.

The state of CBT today

Some educators believe that the field of Computer Assisted Instruction (CAI) is well documented, successful and subject to immutable laws. In fact, the field is in its infancy and its short history is strewn with expensive failures. While some experts try to create a strait-jacket by interweaving Learning Theory and Instructional Design, I believe that there is still a large amount of trail and error experimentation and data collection to be done before any Grand Design will become apparent. The volatility of the field is also enhanced by the constant introduction of new hardware.

AUTHOR provides the subject expert with the ability to directly produce CAI and allows educators to put their theories into practice.

AUTHOR

AUTHOR is an authoring system. Its task is to allow subject experts and educational specialists to develop, present and manage Computer Assisted Instruction. Note that I have not mentioned computer experts, programmers and system analysts. AUTHOR was specifically designed to be used by the author(s) of CAI and not require computer expertise. Of course, this does not mean that AUTHOR users are computer illiterates, most of us now use word processors without being able to program and a certain amount of learning is necessary for effective use the system. I should also be careful not to alienate the computer experts who will wish to use AUTHOR - rest assured that AUTHOR supports a powerful procedural language with a variety of commands to allow programmers to exploit all the power of the system.

The evolution of AUTHOR

AUTHOR sprang from very mundane roots. The precursor to AUTHOR was a limited parser that recognized 3 question types, presented them to students and recorded results. Tests were written using the system text editor (remember line editors?) and results could be examined and sorted in a few limited ways. Nevertheless, this was a major step up from my previous involvement with CAI - developing tests using a general programming language, specifically BASIC-PLUS on a PDP-11 minicomputer. Nor were the tests themselves at the edge of theoretical knowledge - they were simple tests for motor mechanic apprentices. In retrospect, this type of test site was a blessing. We had no Computer Science, Engineering or Business studies depa.tments to effectively monopolize hardware and Computer Centre staff. The student interface had to be simple and robust to deal with the average apprentice who was likely to have no experience of computers at all.

At this stage Microcraft had produced some stand-alone educational software for CP/M microcomputers and Data General minicomputers. Looking for a new project we discussed two possibilities, a wine database for the discerning connoisseur and a generalized authoring system for the teacher. We decided to go ahead with AUTHOR.

At that time I was working in the Computer Centre at Richmond TAFE. Microcraft agreed to provide the software free to the college in exchange for a large test site.

AUTHOR went through five versions at Richmond TAFE before Version 6 was released commercially for the IBM PC. Version 6 supported 5 question types and colour graphics yet required an external text editor and graphics package for lesson creation. Limited student management and results reporting were provided.

Version 7 was released in late 1984 and incorporated a text editor and graphics design system and also some advanced features such as videodisc and speech synthesizer support. Version 8 (late 1986) extended what had become the AUTHOR procedural language and featured a completely redesigned user interface that had both function key and/or pull down menus. Animation and networking were added and the ability to use all the colour graphics and text modes of the IBM PC. I will not enumerate all the additions that occurred over the years but simply refer you to our 750 page manual. I hasten to add that the tutorial section of the manual is a lot more digestible at around 80 pages.

Lesson development

Using AUTHOR is better demonstrated than described but I will attempt to cover the basic method.

1. Plan the lesson or test using the built-in word processor.
2. Use the other editors in the system to create required graphics and/or animations or recorded playback of external software.
3. Expand your p' n into the lesson using the appropriate commands and question types.
4. Test the lesson to see how it will appear to the student.
5. Repeat steps 3 and 4 until you are satisfied with the result.
6. Enter the name of the lesson and a short description into the Lesson register. It may now be run by students.

To alter or update a lesson

1. Call up the word processor make any alterations and test the lesson.
2. Use the other editors to change graphics or animations.

AUTHOR Language

The format of an AUTHOR lesson is quite simple. Any text that is not a command or a comment will be presented to the student. Questions are restricted to 8 distinct types that each have their own format. Various commands may be used to extensively customize the lesson. Branching is usually performed using IF.. THEN.. ELSE statements but simple branching between lessons and randomization may also be used.

The AUTHOR language currently has over 80 commands. Each command must be preceded by a backslash (\). This makes commands easy to distinguish from text in the lesson and also makes the parser's job (and the programming) much easier. The structure may become clearer with the following example

A multiple choice question which will provide feedback if answered incorrectly and then loop until the correct answer is given.

```
\-ANSWERS
\Beginning:
\MULTIPLE
4,1,C

If you specify a command in an AUTHOR
lesson but omit the backslash (\) AUTHOR
will

a. Give up
b. Act on the command
c. Place the command on the screen as text
d. Crash the system
_______________________
\IF WRONG THEN

I'm afraid not! If you omit the backslash the
command will be treated as text and dis-
played on the screen.

\WAIT
\GOTO Beginning

\ENDIF
```

Explanation

The first few lines are comments and will be ignored by AUTHOR. Comments begin with a single quote.

The first command (\-ANSWERS) turns off the automatic display of the correct answer. The next (\Beginning:) is simply a label that marks this point in the lesson.

Next a multiple choice question is specified (\MULTIPLE). The next line specifies 4 choices and 1 answer required of the student, the answer C. You may, of course, have multiple answers.

The question text is finished with a row of underlines called an End Line within the AUTHOR system. The End Line is used by a number of AUTHOR commands to delineate the end of variable amounts of information.

Feedback is given to the student for incorrectly answering the question using a multiple line \IF.. THEN statement. Within this block, text is specified to be displayed on the screen and the lesson will WAIT until a key is pressed. The lesson will then return to the Beginning label (\GOTO) and ask the question again. This process will repeat until the correct answer is given.

When we were designing the language we did go through some mental torment as to whether to include a \GOTO statement. However, although we fully support the concept of structured programming we found that the simple GOTO was more understandable than the other structures of procedures and \WHILE... \WENT loops. In the final analysis we were more interested in subject experts quickly producing effective CAI than producing structured programmers - even at the risk that the structure of some lessons might resemble spaghetti.

Resources

Graphics, animations and colour text screens are considered to be resouces within the AUTHOR system and as such are distinct from the lesson. This allows different lessons to use the same material without wasteful duplication. These resources are created using the appropriate editors.

The graphic editor has all the features of the common PC paint programs and is most effective when used with a mouse, however all functions may also be used from the keyboard. The editor currently supports all CGA and EGA graphic resolutions and the new VGA 256 colour mode will be released in early 1988.

The animation editor allows you to animate up to 4 objects on the screen at once with internal animation of each object.

Design considerations and decisions

- An integrated approach vs stand-alone lessons

An authoring system should be integrated with built-in management of students and results.

A number of authoring systems are similar to general language compilers - i.e. they produce stand—alone lessons that may be distributed. This approach means that lessons cannot be altered and usually exist on floppy disk. The process of integrating these lessons into an easy to use structure is generally left to the user. This is beneficial to the creators of the lesson but not very useful for educators. AUTHOR provides a standard user interface and management of students, lessons and results. Other software may also be defined as lessons within the system and accessed in the same way as AUTHOR lessons.

- Interactive lessons vs production of printed tests

In the early 70's hardware was expensive (especially colour graphics hardware) and a number of systems concentrated on producing printed tests for students which could be subsequently corrected when the student returned to enter his answers into the system. These systems emphasized statistics and had little interactivity. AUTHOR is more concerned with education than testing.

- Use of standard hardware

We deliberately chose IBM PC compatibility because it was obvious, even at an early stage, that it would become the default standard for hardware and software. Education and training departments have never been endowed with large budgets so specialized hardware and expensive software could only appeal to a limited number of people. We did however, require a colour graphics adapter and monitor as we believe them to be essential to the process of CAI. AUTHOR does support more expensive peripherals such as videodisc players and video digitizers, but effective lessons can still be written without the aid of these devices.

- Use of the function keys and/or pull down menu

Functions within the various editors should be rationalised and assigned to common keys. Users should have a choice of interfaces including use of the function keys with the aid of a template and a mouse or keyboard driven point-and-choose menu system. The interface should not clutter the user's work area. Novice users and those with mice tend to prefer to use the pull down menu. The menu can be called at any stage. As soon as the user selects the required function the menu disappears from the work area. Use of the menu is slightly slower than use of direct keystrokes, especially when the user knows the keystrokes without reference to the template. This mode is preferred by experienced users.

- Independence of the user

The system should be directly used by the subject expert with no need for computer experts.

Great care was taken to ensure that lesson and resource creation was straight forward and that the system could be used at a simple or complex level.

- Graphic creation

Graphics should be drawn rather than programmed.

The use of rubber band functions and standard graphic shapes such as circles, boxes, ellipses and lines meant that graphic creation is interactive, fast and possible for those who feel they lack drawing skills.

- Networking and remote delivery

The system should be able to be networked and make remote delivery of lessons possible.

AUTHOR will use any network which operates under DOS 3.+ allowing central storage of student records and lessons. Any workstation on the network may also be used for authoring by instructors with password access.

Remote delivery of lessons is possible in two ways. Standard AUTHOR lessons may be sent on floppy disk or via the telephone system for use with a remote AUTHOR system. Alternatively lessons may be distributed independently of the AUTHOR system using the Lesson System. This system is available to commercial developers.

Conclusion

AUTHOR is currently used in over 140 organizations across Australia and overseas. T ganizations range from universities a lleges to banks and insurance companies. The people using the system have a variety of backgrounds and many have limited experience in using computers. All have used AUTHOR to create effective Computer Assisted Instruction.

Tony McSherry, is one of the directors of AUTHOR, Computer Manager at Richmond College of TAFE and co-creator of AUTHOR. Tony has been working in the area of CAL for approximately eight years. He is also qualified and experienced in the fields of psychology, literature, media studies and video production.

Diverse CBT Applications using AUTHOR

John D. Rundle,
Director Microcraft Pty Ltd

The AUTHOR user community has responded enthusiastically to AUTHOR's flexibility. Five specific lessons are to be demonstrated as developed by AUTHOR users. AUTHOR is used in a wide variety of subject areas.

When we originally designed AUTHOR, our main design criteria were to maximise flexibility and user control of lesson presentation. Our users have responded enthusiastically to this flexibility and have outstripped us in discovering innovative ways to use AUTHOR. When they have been unable to achieve specific goals, we have been badgered into further developing AUTHOR, making it even more flexible.

As part of our contact with our user community, we often view lessons or presentations developed with AUTHOR. Our users have provided us with some excellent ideas and suggestions, some of which are beyond the capability of computers today, but which we will implement when suitable hardware becomes available.

The Applications

Today, I would like to show you five lessons selected from a wide range of available subject areas to illustrate AUTHOR's versatility.

I must stress that these lessons have been developed by our users and while I might suggest ways to improve them, I am presenting them as they have been developed.

None of the developers do not have any special training in CBT, but they are experts in their own subject areas.

The lessons are in everyday use, and have been developed without help from computer specialists, and in most cases without attending specific courses in the use of AUTHOR.

I would also like to thank the authors for giving me permission to demonstrate the following lessons.

- Basic Electrical Circuits. Developed by: Mavis Bird at Richmond College of TAFE. This lesson provides trainee auto electricians with an introduction to electrical circuits.
- Sheep Dentition. Developed by: John Kent at the Melbourne College of Textiles. A lesson covering sheep dentition as related to wool types and the age of sheep.
- Key Facts, A series of lessons on Diabetes. Developed by: Dr Matt Cohen, Director of the Lions International Diabetes Institute Education in self-management for diabetics.
- Chemistry Familiarisation. Developed by: J A Robinson and S Kobot of Queensland Institute of Technology. Familiarisation of students with AUTHOR and CBT.
- Basic Indonesian. Developed by: Jo Fischer, Sacred Heart College, Ballarat. An introduction to basic Indonesian vocabulary.

The Demonstration

I will work through parts of the five lessons as a student, making some deliberate mistakes (and some not so deliberate) so that you can see the lessons as the students see them.

I will also comment on specific features, plus ways of presenting information and asking questions.

Conclusion

Some of the major subject areas covered by AUTHOR users include the following:

Accountancy
Banking
Chemistry
Computer Studies
Computer Training
Diabetes
Drawing
Electronics
Engineering
Internal medicine
Insurance
Languages
Law
Library and information science
Mathematics
Physics
Science
Wool classing

AUTHOR lessons are easily updated and modified and may be transported to any AUTHOR system. Lessons may be sold, exchanged or donated at the author's discretion.

Microcraft Pty Ltd publishes the "AUTHOR Directory: A guide to AUTHOR users and lessons" based on results of surveys of our users. Please contact us if you would like a copy.

Our approach, embodied in AUTHOR, is that the subject experts can develop CBT if given the right tool.

John D. Rundle is one of the directors of Microcraft Pty Ltd, and co-creator of AUTHOR. John's previous experience includes working for Norwich Life Insurance (PC Coordinator), BHP (Information Systems Training Officer) and the Victorian Education Department (Maths and Computer Studies teacher). John is particularly interested in the application of PC's in providing solutions under the direct control of the user.

The Advanced Features of AUT!:!OR

Liz McArthur
Director, Microcraft Pty Ltd

The use of videodisc, robots, gr_phics and speech ,roduction with AUTHOR will be d scribed and demonstrated.

Videodisc

Videodisc technology allows fast, direct access to video and audio information. This makes it very suitable for use with microcomputers for computer based train- It complements the use of the com- :r by offering high quality sound and speech and "real-life" images, both still and moving. The videodisc may also be used to hold over 50,000 still frames making it ideal for picture and graphics storage.

Practical application

Until recently videodisc technology has primarily been used by large organisations such as IBM and GMH which could justify the high cost of production, firstly to create a master and then duplication of the videodiscs.

However, smaller organisations can make cost effective use of videodisc technology by re-authoring existing videodiscs to suit their needs.

AUTHOR offers specific support for the Pioneer LD-V4200 and can be used with any player which can be controlled via the computer's serial port. This means that any CAV (Constant Angular Velocity) disc may be used as a source of still frames and motion picture sequences for several lessons.

Using videodisc with AUTHOR

The \VIDEO command is used in the lesson whenever access to the videodisc is required. This is followed by the commands used by the specific player. For example, the following command for an LD-V4200 player will cause the player to search for frame 5000 and then stop.

```
\VIDEO
FR5000SE
```

Menus to the video material may be created, questions may be asked and sequences reviewed if the student answers a question incorrectly. For example, if the student answers the following question incorrectly the section of the videodisc which deals with this topic is played.

```
\TRUE/FALSE
T
```

An adequate fuel supply is one factor that contributes to efficient combustion.

```
\IF WRONG THEN
```

Unfortunately you are wrong. Strike a key to review efficient combustion.

```
\WAIT
\VIDEO
FR25443SE
26080PL
```

```
\ENDIF
```

Some features:

• Customisation

The \DEFINE VIDEO command is used to assign the functions of a particular player to any of the function keys on the keyboard. For example the following command assigns the play function to the F4 key, still frame reverse to F5, still frame forward to F6, fast speed reverse to F7 and fast speed forward to F8.

```
\DEFINE VIDEO
F4=PL
F5=SR
F6=SF
F7=180SPMR
F8=180SPMF
```

• Text overlay

Text may be overlayed on the video image using AUTHOR if your player (e.g. LD—V4200) has this facility. Text overlay may be used to create titles, arrows and other markers to point out areas of interest and display menus.

• Direct student control

The student may be given direct control of the player from the keyboard by including the \KEYS command in your lesson. You may choose to provide the student with a table of contents on the AUTHOR screen - listing frame numbers with subjects, allowing them to view sections at their own pace and analyse some parts in detail. Functions include:

Play
Reverse
Still frame forward
Still frame reverse
Fast forward and reverse
Frame forward and reverse
Scan
Show frame numbers

The combination of microcomputer and videodisc offers a powerful educational tool. Its use is currently restricted by the cost of production of videodisc material, however, the material already produced can be used in a variety of ways. Two recent developments in the field suggest that the use of videodiscs may increase dramatically over the next few years. These are the ability to store an hour of video material on a Compact Disc and the demonstration of different systems for producing erasable videodiscs.

Robots

AUTHOR lessons may incorporate the control of external devices such as robots for industrial training. Devices may be controlled through the RS-232 serial port from an AUTHOR lesson using the \PORT and \TALK commands.

The \PORT command is used to set up the protocol between the robot and computer. The \TALK command is used to send the robots control language.

Example: controlling the SIR1 robot

```
\TALK
M 0,200
```

This command will move the robots shoulder from position 0 to position 200.

Graphics

AUTHOR supports a range of graphic screens and special effect screen wipes for changes between screens. Graphics may be simply displayed, overlayed with questions or text or presented as a series of slides. The following modes of graphic screens are available:

Text character graphics (created with CED, the Colour Text editor)

40 X 25 character	16 colour	CGA or EGA
80 X 25 character	16 colour	CGA or EGA

Pixel based graphics (created with GED, the Graphic editor)

320 X 200 pixel	4 colour	CGA or EGA
640 X 200 pixel	2 colour	CGA or EGA
640 X 200 pixel	16 colour	EGA
640 X 350 pixel	16 colour	EGA

Special effects

Special effects are used to add interest and continuity to the graphics being displayed and to create the impression of animation.

Twenty four special effects may be used with GED screens ranging from vertical or horizontal wipes to a dissolve. The speed of the effect can be controlled.

Fx	Description
0	Instant load to screen
1	Wipe from top to bottom
2	Wipe from bottom to top
3	Wipe from left to right
4	Wipe from right to left
5	Crush from top & bottom
6	Crush from left & right
7	Expand to top & bottom
8	Expand to left & right
9	Crush
10	Expand
11-20	As in 0-10, but with edges
21	Fade
22	Interleave from left & right
23	Diagonal wipe from bottom. left
24	Diagonal wipe from bottom.right

Video Digitisation using the PC—EYE card.

Video digitisation is the process of converting visual images to digital information. The sophistication of a digitiser is expressed in the number of dots or pixels it can resolve and the number of shades of grey it provides - grey scales. The more grey scales the more life-like the image will appear.

• *Using the PC-EYE*

The image may be captured using a standard camera for input to the graphics editor. The displayed examples use only 4 grey scales at a resolution of 640 x 350 pixels. A video digitiser can save a substantial amount of time in creation of graphics.

• *Practical application*

Best results are achieved with black and white line drawings.

Voice Production

There are two ways of creating speech - digitisation and synthesis. Speech digitisation samples real speech and converts it to digital information. This gives high quality speech but requires a large amount of disk space for storage of the information. Speech synthesis builds up words from phonemes.

• *Using the VOTALKER IB speech synthesizer*

The VOTALKER comes with a standard set of words and the user can also build a set of words which may be modified using the phonetic editor. The rate of speech, amplitude, inflection and filter can be controlled.

The \VOTALKER command is used to send text to the speech synthesiser.

For example.

```
\VOTALKER
2
Hell[I=5] o. [i=0]This is the VOTALKER speech board.
```

The VOTALKER will say this sentence with an upward inflection on the o in Hello after an initial delay of 2 seconds.

- *Practical application*

Realistic speech can only be obtained by patient use of the phonetic editor.

Conclusion

AUTHOR may use a variety of perhipherals as lesson resources. Any device that has an RS232 serial port may be controlled from within an AUTHOR lesson. We are committed to the continuing evolution of the AUTHOR system. As new devices which promise to enhance the learning process appear on the market, they will be incorporated into the AUTHOR system.

Liz McArthur, is one of the directors of Microcraft and co-creator of AUTHOR. Liz is also qualified and experienced in the areas of psychology, history and philosophy of science, librarianship and production of audiovisual media.

Object-Oriented Control for Intelligent Computer Assisted Learning Systems

Tom Richards, Dept of Computer Science, La Trobe University
Geoff Webb, Computing & Information Studies, Griffith University
Noel Craske, EDP Depeartment, Chisholm College

ABSTRACT

This paper investigates an approach to providing a general-purpose authoring/tutoring shell for intelligent computer assisted learning systems. The approach is to outline an object-oriented representation of task/goal hierarchies, then to consider the ways in which domain expertise, student information and teacher expertise can be made to interact with such hierarchies. The result is a skeleton in terms of which some exploratory lessons are being constructed; and in terms of which further research on the domain expert system, student modelling, educational expertise modelling, and user interfacing can more concretely be developed.[1]

Introduction

Most current work on the construction of Intelligent Computer Assisted Learning (ICAL) systems has concentrated on one-topic systems: diagnosing faults in a particular electrical circuit -- the SOPHIE system (Brown, Burton & DeKleer, 1981), solving the WUMPUS game (Goldstein, 1981), and ACE, which discusses the analysis of nuclear magnetic resonance spectra (Sleeman & Hendley, 1981). These systems and others like them were not intended as vehicles for widespread educational use; rather, they are vehicles for investigating and solving important research problems: natural-language communication, modelling the evolution of procedural knowledge in a learner, and dialogues to monitor and guide student reasoning, respectively. Those are

[1]This research is supported by grants from the Apple Educational Foundation, Australian Telecommunications and Electronics Research Board, La Trobe University School of Mathematical and Information Sciences, and the Victorian Department of Industry, Technology and Resources.

problems that need solving first if we are to use artificial intelligence effectively in education.

In the EXCALIBUR Project, by way of contrast, the authors and others are investigating what is involved in providing a *general-purpose* ICAL system, one which provides an authoring and knowledge engineering shell, analogously to a standard expert system shell, and a delivery system for students (Richards, 1986a). Such a system is intended to be general-purpose in that the architecture and knowledge representation methods used should involve no specific assumptions about the content or nature of the subject domain to be taught, nor about the modes of learning or presentation (instructional, student-driven, exploratory, problem-oriented, etc) to be used.

To use such a system, a person whom we will call the *author* will interact with the system to create what for lack of a less perjorative word we will call a *tutorial* — a program which can then be run by a student with the aim of learning from his or her interaction with it. We stress that it is up to the author to decide the extent to which the student's interaction with the tutorial is to be constrained by (for instance) predetermined delivery sequence and student assessment results, in which case the author will need to specify instructional modes or rules to the system; or to which the student may treat the tutorial as an educational information kit and tool bag, in which case the author will need to specify what resources and modes of exploration are available to the student. The point is that a general-purpose ICAL system should place little constraint on the educational philosophy the author wishes to adopt.

The question is: how might one build such a general-purpose system? The main problems we have identified, for which solutions must be found that do not compromise the sought-after generality, are:

(a) How to store data on the subject domain of the tutorials, and how to perform analyses on that material in ways relevant to the learning process at hand (designing the *Domain Information System* facility, which in many cases will amount to an Expert System or Knowledge Based System);

(b) How to construct a knowledge base about the student, including what the student believes and has learnt in the domain of the tutorial, and why and how. This facility is called the *Student Model*.

(c) How to incorporate rules about student knowledge and performance criteria that can be used to tailor or guide the presentation of the student session in the light of information gathered about the student so far. This is the *Mentor* facility (Sussex, 1987).

(d) How to unify these under a *System Driver* which handles authoring sessions as well as student sessions.

The Domain Information System is not necessarily the same as the material which the student may learn. Often, it may be much more extensive, for the same reason as a teacher usually needs to know far more about a subject than s/he will teach, just in order to handle the teaching process well. In the case of an ICAL system, the Domain Information System may often need to contain intelligent subsystems enabling the computer to solve problems whose solution is needed for the tutorial interaction, but whose solution method is either far too advanced for the student or is a computer-oriented process that can be quite confusing to a human. Crudely put, machines and people do not think alike. Examples are:

- The parsing of a given French sentence in a French grammar tutorial, in order to generate the specific grammatical and pragmatic information the student needs for the interaction about that sentence. Machine parsing methods are usually totally different from those used in a French language tutorial class, even if the end-products (parts of speech, agreement relations, etc.) are much the same.
- The checking or production of a proof step in the derivation of a proof in a tutorial on mathematics or formal logic. A student may, for example, be learning one particular method of doing formal logic, such as natural deduction; yet the tutorial system will use a pattern-matching procedure which is not in the least educationally relevant, to check the student's proof steps.

We cannot easily do without the special expertise of the Domain Information System. If we did away with the French parser, then an author could provide the parsing information needed by the student for each stored sentence; but then that information could not be generalized to cover sentences typed in by the student. In the logic example things are worse, for proof checking information cannot in principle be provided by the author: for non-trivial derivations there can be an infinity of correct "next steps". Only a special proof-checking procedure will do the job.

We turn now from considerations about the material to be learned, to questions of how to learn it. In French grammar, logic, and indeed courses on most topics there may be many modes of learning. The student might be, for example:

- Working through a set system-controlled sequence of actions,
- Freely exploring an environment containing knowledge and techniques relevant to the subject domain,
- Trying to devise his/her own techniques for reaching a goal, and testing them.

The more of such learning modes as these that are made available to the student, the more demands are put onto the content and problem solving power of the underlying Domain Information System. But even more stress can be put onto the other three facilities, the Student Model, the Mentor, and the System Driver.

In the rest of this paper we will look at a method of designing and unifying these four facilities. The result is the specification of an architecture for a general-purpose ICAL system.

Goal/Task hierarchies

One place to start in trying to analyze this problem is the idea of*Goal/Task hierarchies*. See Figure 1.

A tree-structured Goal/Task hierarchy has at its root the overall goal of the current tutorial. and the (immediate) dependants of any node are the subgoals that need to be achieved in order to achieve the node's own goal. On each node is a task, which to be carried out requires that the the tasks on the dependant nodes have been executed. The terminals of the tree are initial goals which have no prior goal: tasks which can be executed without performing prior tasks. Formally. the links from a node N to its dependant nodes N_1, N_2, ... mean that the tasks on the dependants are individually necessary and jointly sufficient for the execution of the task on N; so a task hierarchy is an and-tree. and a link from N to a dependant node N_1 means that achieving the goal on N_1 is a necessary condition of achieving that on N. That is to say, in order to achieve the goal on N it is (logically, pedagogically. epistemically, or ...) necessary to achieve. or to have achieved. the goals on N_1, N_2....; and when they have been achieved it is (logically, pedagogically. ...) possible to achieve the goal on N by doing the task on N. The relation between the goal and the task on a node is simply that the goal is what has been achieved when that task has been executed. given that the subgoals on N_1, N_2. ... have been achieved. Any tighter definition of the linkage relation in a goal/task hierarchy is not really a matter for the system designer. but for the author of a tutorial.

who will wish to exploit it as a control structure that may be adapted to his or her ideas about the flow of the tutorial.

In Figure 2 we have a small example of a Goal/Task hierarchy for detecting subject-verb agreement in number, which illustrates the points just made.

The knowledge needed to determine agreement of subject and verb (the goal on N) is the gender and number of the subject (left-hand sub-goal) and the gender and number of the verb (other sub-goal). When they are known, a comparison of them to see if they are the same (task on N) suffices to reach the goal on N.

Suppose now a student is attempting to use an ICAL program based on this Goal/Task hierarchy to learn about subject-verb agreement. We imagine that the student is presented with a series of examples -- French sentences that may or may not obey the rules about agreement — and is required to move through the hierarchy with each example. (The pedagogical effectiveness of this approach is not up for discussion at present: we simply need an example to motivate this description of our system.) Then on each node we need to record a method of determining if the student is correct in his/her task solution. One way to do this might be to look up the answer as entered by the tutorial author on a record containing the current sentence example (S1 of Figure 3 is a record containing the needed data). More intelligently, such a frame might be created by a parse procedure belonging to the underlying Domain Information System (Figure 4).

Figure 3 contains (sketchily) three types of control (or, if you prefer, information).

1) Domain information, as in the record S1.

2) Tutorial driving procedures, as in the StudentTest slot of N1.

3) Student modelling services, as in the StudentRecord slot of the frame N1.

Each of these three control regimes may embody a great deal of information, procedures, and the like: but as Figure 3 suggests, each regime intersects with a Goal/Task record, such as N1, in a simple way. The architectural design of the system is based on this, brings us to the central design issues of the system.

Object hierarchies and Type-systems

In Object-Oriented Programming Languages (OOPL's: see Byte, 1986; OOPSLA, 1986) the entity that does the work is not a procedure, as in procedural languages, but an Object (with a capital O to avoid ambiguity) containing local data (like Pascal records) and procedures (called Methods), which are accessed and activated by Messages (like command lines) sent by one Object to another, and returning values. Objects are similar to the sets and individuals of set theory, in that one Object can be a member (usually called an instance) of other Objects; or it can be a subset (subtype) of other Objects. For instance, an Object containing information about the behaviour of the French verb *voir* will be an instance of another Object containing generic information about the behaviour of transitive verbs. That Object in turn will be a subtype of another Object containing generic information about the behaviour of verbs in general. Note the difference between an instance and a subtype: *voir* is an instance of a transitive verb, and of a verb. It is not a subtype of either. And a transitive verbs, as a type, are not an instance of verbs, they are a type of verb.

Subtype relations form a hierarchy (see Figure 5), with instances as the terminal nodes of the hierarchy tree.

Objects in the tree normally inherit properties belonging to any higher Object; for instance if Verb in Figure 5 recorded that all verbs have subjects, that would hold of transitive verbs and intransitive verbs, and all instances *être* and *voir* as well. So if a query were addressed to this tree: 'Do transitive verbs take subjects?' the TransitiveVerb Object would receive this query message and answer 'Yes.' If the lesson designer wished to treat the verb *être* as an exception to the rule about subjects, s/he records on the Etre object that it does not take a subject, and the inheritance of the contradictory property from Verb is blocked. The query 'Does *être* take a subject?' would go to the Etre Object, which would directly answer 'No,' without consulting further up the hierarchy. In the same way procedures can be stored on the appropriate Object, and get consulted by lower objects unless overriden by a lower procedure. The procedure for finding a verb's subject belongs on Verb, and the procedure for finding a verb's object belongs on TransitiveVerb.

Objects that are instances typically record different sort of information from objects that are types. Instance Objects record information specific to the thing that Object represents, but a type Object will record generic information true of all instances of that type. The type Object StudentModel would record the pass mark needed for any student to succeed in the current tutorial; but an instance of that type. such as Student36, would record that student's name and mark achieved.

In our ICAL design, the Goal/Task hierarchies are not Object hierarchies as described above, even though the nodes (see Figure 2 and N1 of Figure 3) are Objects. So from now on we will call them Goal/Task Objects. The hierarchy shown in Figure 2 is plainly not one of inheritance, for N1 does not inherit anything from N. The link as remarked earlier, is that the lowerGoal/Task Object contains a goal that is a necessary condition for attaining that in the higher Object. A goal. filling a slot in a Goal/Task Object such as N1, is however an instance of the type Object called Goal, which specifies the structure of any of its instances, which are of course individual goals. (See Figure 6.)

Figure 6a shows a type Object called Goal containing a number of structured InstanceSlots, which describe generically the slots occurring on each token of this Object that may get created. i.e. on any individual goal Object. Goal5 is a token of Goal; and what the Goal slot of N1 in Figure 3 really contains is a pointer to Goal5. To create Goal5. the InstanceSlots of Goal are called upon to define the structure of Goal5. and so is any Object of which Goal is a subtype -- in this case Universe. which is the unique upper bound of the inheritance lattice. This is the method used to create any new Object.

The OwnSlots contain data or Methods belonging to that Object itself; these slots are the ones defined by the corresponding generic InstanceSlot data on the higher Objects. One Object can access the data in another Object's OwnSlots, or activate Methods stored there. by sending it a Message, containing arguments for the Method's parameters if needed. For example, since the Goal slot of a Goal/Task Object points to some instance of Goal, Goal/Task Objects are designed so that they can send messages to Goal. For example. the pointers to Goal/Task nodes making up the Goal/Task hierarchy (as seen in Figure 2) are determined by a Goal/Task Object sending a message to the instance of Goal that its Goal slot points to. In this way, a Message from N1 to its Goal instance G5. to find G5's supergoal. will return N's goal. Then a Message to that goal to report its subgoals will return G5's "sibling subgoals" (just one, in the example) which must also be achieved by the student before the goal of N can be attempted. Messages can now be sent from N1 to each of those subgoals. accessing the Goal/Task objects they belong to (just N2) and checking which of them the student has already achieved. This helps to determine the flow of tutorial control.

Where does the code for these messages reside? In OwnSlots in the type Objects Goal/Task and Goal, to be inherited by their instances. For example, the Method to find a Goal/Task Object's siblings resides on the type Object Goal/Task, and looks like this, in pseudocode:

```
GetSiblings:
        supergoal := ask Goal for its supergoal Object;
        sib-goals   := ask supergoal for list of its subgoal Objects;
        siblings    := collect into a list:
                            for each sib in sib-goals:
                                ask sib for its Goal/Task Object;
        return (siblings less self).
```

Now if N1 is sent the message GetSiblings it will return the list (N2).

This example of how a slot in an Object can point to another Object, which belongs to a particular hierarchy of Objects, illustrates an important aspect of the architecture of our design. We have identified a number of Object hierarchies, or clusters thereof, that are needed in our ICAL structure[2]. Major ones are:

- Goal type-systems, discussed in detail above.
- Task type-systems, defining activities for the student.
- A type-system to describe the Student Models for individual students and associated diagnostic procedures.
- Logic type-system to provide rule frameworks for a logic-programming reasoning service where needed. Sets of these rules comprise the Domain Expert System, reasoners about the student's belief base, and the analysers for a natural language interface.

Object networks

We remarked earlier that the hierarchy of Goal/Task Objects was not linked up by inheritance, so in present terminology it is not a type-system. Type-systems are used to define Objects as types, subtypes, and ultimately instances, and to provide through inheritance powerful built-in reasoning services. But the hierarchy of Goal/Task Objects is effectively a simple semantic network: data nodes joined by labelled links.

[2]A *type system* here is a sublattice of the one overall Object lattice, and it has a single upper bound which is some very significant type Object such as Goal or Student or LogicRule.

Overall, the system we are designing can be viewed as comprising several connected semantic networks or conceptual graphs (Sowa, 1984), which we will call Object networks. Major Object networks in a tutorial program are:

- The underlying Domain Information System
- The Student Model facility
- The Mentor facility
- The System Driver
- The Goal/Task network
- Human/machine interfaces
- Natural language processor
- Reasoning systems, primarily for the Domain Information System if it is knowledge-based, and for conducting rational dialogues with system users.

Concentrating attention on these Object networks as the architectural framework of an ICAL system makes the type-systems disappear into the background. On the other hand, if you concentrate on the type-systems, then you see the Object networks as largely derivative structures, being created by the pointers in slots of the Objects as created, maintained, and manipulated by the type-systems. Look at how we found the siblings of a particular Goal/Task Object, which illustrates how the Goal type-system defined the Goal/Task Object network.

Figure 7 tries to depict, on two-dimensional paper, the multi-dimensional structure of the resulting system.

It would be better to visualise slots (in Target of Figure 7, for instance, not as containing pointers, but containing the Objects pointed to, and to visualise each type-system as lying in a different plar or dimension. The Object networks cannot be entirely determined by the background type-systems and slot pointers; but to the extent that they are, the authoring task for a given tutorial is simplified and automated. The author is presented with ready-made prototypes for type-systems, e.g. the type Objects Goal/Task and Goal, and needs to specify instances and what goes into the OwnSlots of those instances. That can be more like a data-entry task than a programming task. To the extent that Methods need to be developed by the author of a tutorial, typically in an intelligent Domain Information System, then to that extent programming or knowledge engineering is needed.

At its simplest, then, the tutorial construction procedure for an author is:

- Construct a Goal/Task network for the tutorial domain.
- Construct a Domain Information System which will provide the domain data and processing that a student (or the system) may need at each node of the Goal/Task network.

From the point of view of the system designer, who has to think about how to make the authoring process simple yet powerful. three important research goals that we are exploring are:

- To devise skeleton Goal/Task node objects that put few constraints on the type, content, or style of tutorial to be developed around those tasks, yet which allow the author to express the cognitive structure of the tutorial with a minimum of special programming or modification.
- To provide a clean interface to the Domain Information System that allows domain information to be entered securely; and in a way that permits reasoning on that information and creates suitable output for student users.
- To develop an author/system interface that requires as little learning as possible by authors. while giving them as much power and control as possible over the system and the tutorial they are creating. To do this very high-level editors and browsers are needed. such as network displays and network editors.

To illustrate the problem of designing authoring facilities for a general-purpose ICAL, and the way in which an Object-based architecture helps, we will now look at the question of providing various tutoring strategies or methods for any given tutorial material.

Provision of Tutorial Strategies

The object-oriented approach allows several useful features and procedures to be developed. virtually free. as aspects of the OOPL methodology.

- The Goal/Task hierarchy can be generalised to an and/or digraph, in the manner of the Feature Networks of (Webb, 1986). This now allows us to represent any structure of necessary and sufficient conditions between nodes, which give great freedom in representing the epistemological structure of the concepts, skills, etc. being tutored.
- DoFirst and (inversely) DoNext slots on a Goal/Task Object can contain pointers to other such Goa/Task Objects, thus controlling lesson sequencing in a rigid manner. Alternatively, the Goal/Task hierarchy can be used to provide default tutorial sequencing. If the type Object for these nodes has an instruction to do first the dependants of a node (if any), then a bottom-up tutorial regime is established as the default. Or, an instruction there to do the parent first establishes a top-down regime, allowing the student to avoid doing explicitly the subtasks of a task s/he can do in one hit.
- A TutorialStrategy slot on a Goal/Task Object can contain code for an explicit tutorial strategy, to override the contents of DoFirst and DoNext slots, but accessing them if needed typically under certain conditions laid down by the author or inferred from the student model.
- DoFirst and DoNext slots will in general create a pointer universe for the Goal/Task Objects that organizes them as a directed lattice or even a general digraph, with lower bounds and upper bounds representing the first and last things to do. This can be even more complicated if these slots contain disjunctions or conjunctions of pointers. A system-provided graph search algorithm will act as a tutorial planner for the student, finding feasible paths to the root task/goal node(s).
- In general, a student will have some task to do at a Goal/Task node, even if it is only to read something. Slots on the node can specify different tasks for different tutorial strategies, or for different conditions that the student is in: slots such as Remediate, CheckKnowledge, TestKnowledge, TellStudent. Access to these will typically be determined by the code on the TutorialStrategy slot. Thus, except in rather primitive lessons, the Goal/Task network need not of itself define any sort of sequencing of student tasks.
- The task(s) at a Goal/Task node can recursively be defined in terms of another "lower-level" Goal/Task hierarchy.

effectively providing different levels of abstraction or detail at which a student might proceed with the tutorial.

- Costs can be statically or dynamically associated with nodes on the graph, reflecting difficulty of task, desirability of doing the task, etc. The graph-search algorithm mentioned above will then seek minimal-cost paths for the student.

- Any slot on a Goal/Task node can have a UserAccess? facet on it, indicating whether the user may access the information or execute the procedure in that slot. This provides a control over the openness of the tutorial material to the student, and the extent to which they can browse, select their own learning method, etc. Such a facet may of course contain code, not just Yes or No, determining the access conditions for that slot, and distinguishing between types of user, such as programmer, tutorial author, and student.

- Authoring a tutorial can be treated as one mode of use of the tutorial. This allows a student access to the authoring tools, and a high level of access to slots in the Goal/Task network, as well as to the Domain Information System and the Student Model system. This has the advantages of providing a uniform user interface, and not one for the author and another for the student; and of opening up a powerful learning-by-making tutorial strategy. The disadvantage is that the student could damage or destroy the tutorial by his/her modifications; so a student-as-author access class needs to be distinguished from a true-author access class.

Student Modelling

Here too, the OOPL approach to system design allows a lot of student information to be collected and used more or less automatically.

- Demons on a Goal/Task node can collect any type of monitoring data on student behavior when the goal is visited. The data can then be dispatched to appropriate nodes in a student-modelling structure. Demons on nodes in that structure can then update student statistical data, revise the profile of the student, and despatch information to the blackboard that will be needed for the next stages of user interaction

- Collected student data, either on a Goal/Task node or in the Student Model database or on the blackboard, can be accessed by message-passing to help control the tutorial or to determine how to respond to the user.

- Overlay models of student beliefs (Goldstein, 1981) are effectively generated automatically by storing student beliefs about a Goal/Task node on that node, and similarly by storing student beliefs about some item or concept in the Domain Information System database on the Object encoding that concept. This provides the raw data for analysing bugs in student beliefs (Stevens, Collins & Goldin, 1981), which are important in directing further tutorial interaction.

Comparison with other systems

The idea of Goal/Task nodes is found in a simplified form in the ECCLES system (Richards & Webb, 1985); although there they are arranged strictly in an or-tree whereas here any and-or graph is constructible. The point of the or-tree was that a path through the tree represented an appropriate series of tasks for the student when faced with a particular problem in the domain. Analysis of the grammar of one type of French sentence, for example, would proceed down one path, determined by the grammatical features of that type, and analysis of sentences of another type would proceed down a second path, taking alternatives appropriate to the different features of the second type. Each sentence to be analysed is flagged by a tip node in the decision tree, thus determining the path of tasks to take. See Figure 8.

Also, in ECCLES, the actual tutorial interactions are determined by question-response material stored on each node. On the present model, many tutorial modes are possible, and a student is not constrained to any path through the Goal/Task network (though can be). Moreover, in the present system, if a student is "at" a particular Goal/Task node, the interaction there can be of many sorts, and be determined by processes in the Student Model, Mentor, and Domain Information System, and as outlined in the previous section of this paper. Teaching material on a Goal/Task node generally just consists of pointers to the relevant objects in the Domain Information System, or, in more complex cases, messages to the Mentor who organizes appropriate material from the Domain Information System. It is important to realize that the Goal/Task Objects in the present system are mainly there as points of intersection of Objects from many type-systems, thereby acting as nodes for an Object network (the Goal/Task network) that describes epistemological relationships in the subject domain to be taught.

The Goal/Task network provides a description of the operations that students must perform to complete learning tasks from a subject domain. Often this network will map directly onto the Domain Information Ssytem -- the network that describes the knowledge that the student is to acquire through the performance of those tasks. For example, in Figure 2, N_1 should map directly onto a node in the Domain Information System specifying that the number of the subject of a sentence must be singular or plural. N_2 should map onto a similar node relating to verbs. Finally, N should map onto a Domain Information System node specifying that there is subject/verb number agreement in a phrase if and only if the number of the subject is the same as the number of the verb.

A Goal/Task network will only map directly in this manner onto a Domain Information System network if each element of knowledge that the student is to learn is to be taught by a single operation that the student is to perform and if each operation in turn relates only to a single element of domain knowledge. There will be many domains for which these conditions do not hold.

The idea of basing lessons on explicit descriptions of the knowledge to be taught has been explored through the DABIS system (Webb, 1986). DABIS utilises a form of and/or digraph, the feature network, to represent the epistemological structure of the domain to be taught. Thus, DABIS lessons are based on explicit descriptions of the knowledge they are to teach (the Domain Information System) rather than on explicit descriptions of the teaching strategy that is to be pursued (the Goal/Task Hierarchy), as is the case with ECCLES. Figure 9 illustrates the difference. This allows DABIS to construct far better student models than ECCLES in domains for which there is not a one to one correspondence between the Goal/Task hierarchy and the epistemological structure of the domain.

However, DABIS has the shortcoming that it is difficult to specify instructional flow that differs substantially from the structure of the Domain Information System. Thus, like ECCLES, and for the converse of the same reason, DABIS operates best in domains in which the Goal/Task hierarchy maps directly onto the Domain Information System. As a result, tutorials for both systems tend to be selected where these conditions hold. This, unfortunately, tends to obscure the fundamental differences between the semantics of the underlying structures utilised by the two systems. However, as described in (Webb, 1986), these differences are both real and significant.

The present Project, EXCALIBUR, can be seen as unifying the ECCLES and DABIS approaches to allowing the explicit description of both a Goal/Task network and a Domain Information System within the one system. What EXCALIBUR adds to the other systems is:

- explicit Goal/Task networks (missing from DABIS),
- explicit knowledge structures (missing from ECCLES),
- the allowance that student paths through the Goal/Task network need not be constrained by the necessary-condition vectors of that network. but by pointers in DoFirst and NoNext slots in the Goal/Task Objects,
- the provision an Object-oriented architecture for the construction of ICAL's around the Goal/Task networks.

The reason for treating the Domain Information System separately is just that in general a lot more information and reasoning ability is needed in a teacher than the curriculum they teach (see the Introduction, above.) However, the Domain Information System can, in the Object-structured architecture of the present system, be coupled as closely as needed with the Goal/Task network. In some cases it could be dispensed with entirely; as where the tutorial author provides the necessary parse data for the problem French sentences.

Conclusion: further research

The design of a suitable architecture is just the starting point of the EXCALIBUR Project. Current active research by the authors consists of:

- Building small tutorial systems along the above lines, as different in subject area and tutorial approach as possible, as testbeds for the design of Goal/Task hierarchies and nodes. and of the various Object networks and their interaction (Richards, 1986b)
- Abstracting from those cases to an adequate common Object formalism.

Other current EXCALIBUR work comprises:

- Design of Student Model systems that incorporate different theories of cognitive structure and learning; with the aim of using EXCALIBUR as empirical tests of those theories (Cumming, 1986). The use of ICAL systems as testbeds for cognitive theories has been explored by (Anderson *et al*, 1984).

- The problem of incorporating current work on "Instructional Design" – the study of how to design good teaching vehicles and learning environments -- into EXCALIBUR as features available by default or on the author's choice (Barrett, 1986).
- Natural language processing for the system (Sussex, 1987).
- User-computer discourse management, including answer negotiation, logic of dialogue, and modelling student beliefs (Girle, 1986.)[3]

REFERENCES

Anderson, J.R., Boyle, C.F., Farrell, R., & Reiser, B.J. (1984). Cognitive Principles in the Design of Computer Tutors. *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*. Boulder, Colorado.

Barrett, J. (1986). Intelligent CAL -- some development considerations. *Proceedings, 1986 CALITE conference*. University of Adelaide. 23-37.

Brown, J.S., Burton, R.R., DeKleer, J. (1981). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In (Sleeman & Brown,1982), Ch.11.

Byte, (1986). [Special issue on OOPL's] *Byte*, 11:8, August.

Cumming, G. (1986). The Mentor Component in EXCAL. In *Proceedings of the First Round Table Conference*, La Trobe University, EXCALIBUR Project, Technical Report 3.

Girle, R.A. (1986). Dialogue and discourse. *Proceedings, 1986 CALITE conference*. University of Adelaide, 123-135

Goldstein, I.F. (1981). The genetic graph, a representation for the evolution of procedural knowledge. In (Sleeman & Brown, 1982), Ch.3.

[3] The authors wish to thank Jan Reed, Dennis Hill and Roly Sussex for valuable discussions.

OOPSLA (1986). *Proceedings of the OOPSLA '86 Conference.* Constituting *SIGPLAN Notices* , September 1986.

Richards, T.J. (1986a). A Description of the Educational Expert Systems Project. La Trobe University, EXCALIBUR Project, Technical Report 1.

Richards, T.J. (1986b). Knowledge representation in expert system based computer assisted learning. *Proceedings, 1st Australian Artificial Intelligence Congress* , Deakin University, section H.

Richards, T.J., Webb, G.I. (1985). ECCLES - An 'Expert System' for CAL. *Proceedings of the 1985 Western Educational Computing Conference* Oakland. California. California Educational Computing Consortium, 151-157.

Sleeman, D., Brown, J.S. (eds) (1982). *Intelligent Tutoring Systems.* London, Academic Press.

Sleeman, D.H., Hendley, R.J. (1981). ACE: a system which analyses complex explanations. In (Sleeman & Brown. 1982), Ch. 5.

Sowa, J.F. (1984). *Conceptual Structures.* Reading, MA, Addison-Wesley.

Stevens. A., Collins, A., Goldin, S.E. (1981). Misconceptions in students' understanding. In (Sleeman & Brown, 1982), Chapter 1.

Sussex. R. (1987). Natural Language Interfaces and Representations. La Trobe University, EXCALIBUR Project, Technical Report 4.

Webb. G.I. (1986). *Knowledge representation in computer-aided learning.* PhD. Thesis, La Trobe University, Department of Computer Science

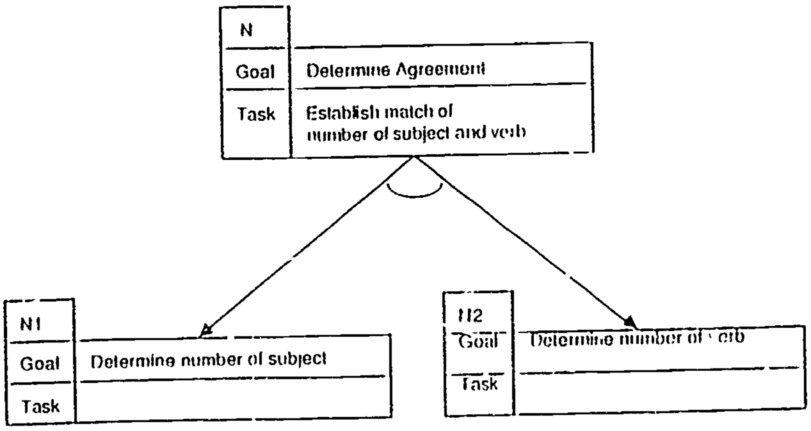

N
Goal
Determine Agreement
Task
Establish match of number of subject and verb
N1
Goal
Determine number of subject
Task
N2
Goal
Determine number of verb
Task

Figure 2

A task/goal hierarchy for subject verb agreement

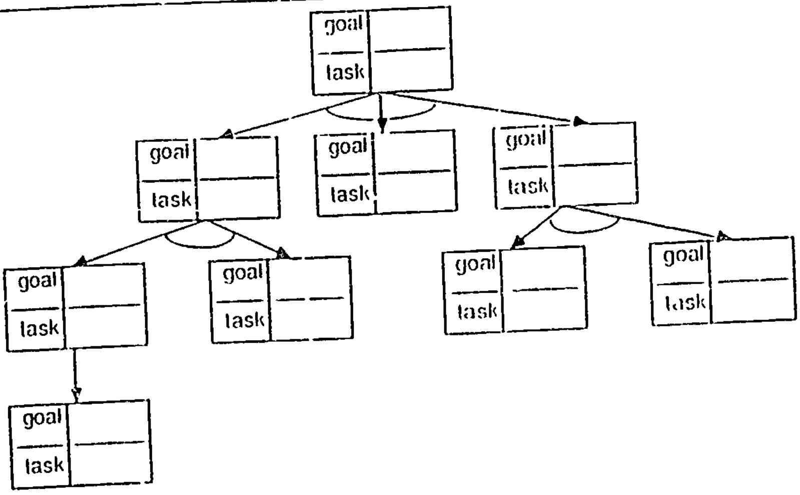


Figure 1

A tree-structured Goal/Task hierarchy

S1

OwnSlot	aSentence
Sentence	The girls were running
Parse	*parse data*
Subject-number	Plural
Verb-number	Plural
Subject	The girls
Verb	were running

N1

Goal	Determine number of subject
StudentTest	(= (response-to "What is the number of the subject of this sentence") (subject-number CurrentSentence))
StudentRecord	(if (= StudentTest true) then (put CurrentStudent Goal 'passed) else (put CurrentStudent Goal 'failed))

Figure 3

A Record or Object S1 holding an example sentence and grammatical information about it, and a corresponding Goal/Task Object N1. Italic text is a description of data, not the data itself. Parenthesised text is Method code

SENTENCE

InstanceSlot	aSentence
Sentence	
Parse	*parsing function returning parse data*
Subject-number	(subject-number Parse)
Verb-number	(verb-number Parse)
Subject	(Subject Parse)
Verb	(Verb Parse)

Figure 4

A type Object for sentence data Note that S1 of Figure 3 is an instance of this Object.

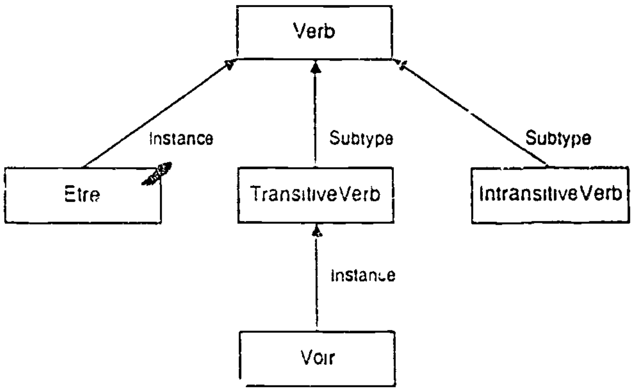


Figure 5

An inheritance hierarchy of Objects

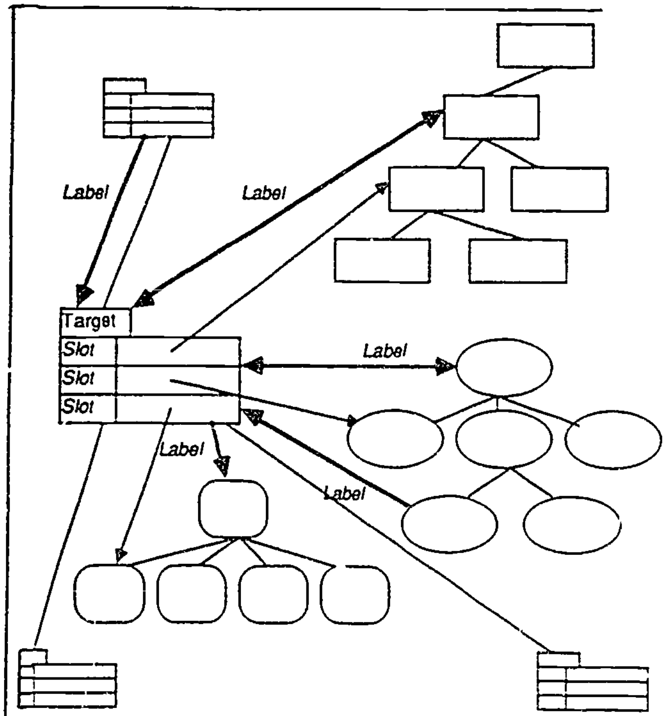


Figure 7

Type-systems and Object networks

Four type-systems are shown, linked by plain lines showing inheritance. One Object, called Target, shows its slot pointers (thin arrows) to other Objects. Inheritance and slot pointers (those for other Objects are not shown) allow Methods in Objects to determine the conceptual graph relations (heavy arrows) that define an Object system

(a)

Goal		
InstanceOf		
SubtypeOf	Universe	
InstanceSlot	theGoal	
	Key	*function returning unique keys*
	Description	*instances put English description here*
InstanceSlot	Goal/Task	*instances put a pointer to a Goal/Task Object here*
InstanceSlot	Subgoals	*instances put pointers to subgoals here*
InstanceSlot	Supergoals	*instances put a pointer to the supergoal here*
InstanceSlot	GoalData	*instances put Methods or pointers here to obtain data about the goal*

(b)

Goal5		
InstanceOf	Goal	
SubtypeOf		
OwnSlot	theGoal	
	Key	Goal5
	Description	Determine number of subject
OwnSlot	Goal/Task	N1
OwnSlot	Subgoals	()
OwnSlot	Supergoals	Goal49
OwnSlot	GoalData	()

Figure 6

(a) The type Object "Goal" (simplified);
(b) An instance, Goal5, of Goal.

476

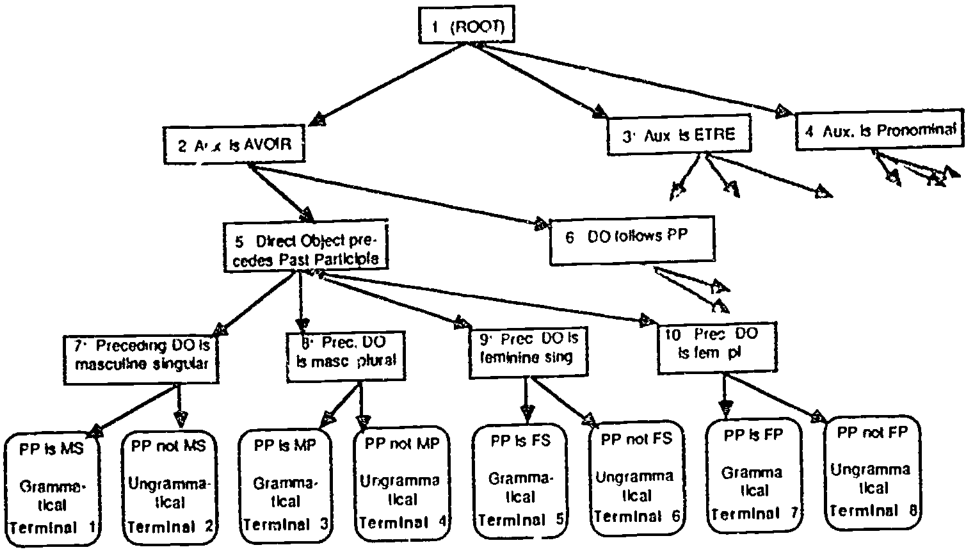
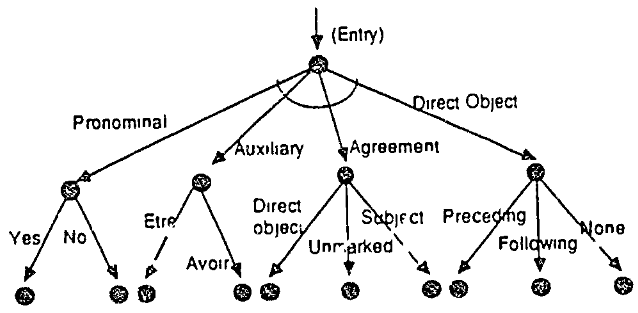

1 (ROOT)
2 Aux is AVOIR
3 Aux is ETRE
4 Aux. is Pronominal
5 Direct Object precedes Past Participle
6 DO follows PP
7 Preceding DO is masculine singular
8 Prec. DO is masc plural
9 Prec DO is feminine sing
10 Prec DO is fem pl
PP is MS Grammatical Terminal 1
PP not MS Ungrammatical Terminal 2
PP is MP Grammatical Terminal 3
PP not MP Ungrammatical Terminal 4
PP is FS Grammatical Terminal 5
PP not FS Ungrammatical Terminal 6
PP is FP Grammatical Terminal 7
PP not FP Ungrammatical Terminal 8

Figure 8

Part of an ECCLES decision tree for analyzing the French *passé composé* construction Nodes show the information gained to reach them; terminal nodes (rounded boxes) also show the path number to that node.



55

Figure 9

Part of a DABIS feature network for the French *passé composé* This covers much the same features as the ECCLES tree in Figure 8. The loop joining the links coming down from the top node indicates that it is an AND-node: all four subtrees below it must be visited, not just one.